1992

A fatigue damage monitoring system for large structures

Qinghong Cao
Lehigh University

AUTHOR: Cao, Qinghong

TITLE:

A Fatigue Damage Monitoring System for Large Structures

DATE: May 31, 1992

A FATIGUE DAMAGE MONITORING SYSTEM FOR LARGE STRUCTURES

by

Qinghong Cao

A Thesis

Presented to the Graduate Committee

of Lehigh University

in candidacy for the degree of

Master of Science in Electrical Engineering

Lehigh University

Bethlehem, Pennsylvania

1992

This thesis is accepted in partial fulfillments of the requirements for the degree of Master of Science in Electrical Engineering

May 11, 1992
(date)

Weiping Li, Ph.D.

Kenneth K. Tzeng, Ph.D.

Acknowledgements

I wish to express my sincere gratitude to Dr. Weiping Li, whose advice, support and encouragement have helped me through all phases of this project. I have learned a lot about the principles and applications of VLSI digital signal processing from him. I am grateful to Dr. Ben T. Yen , Dr. Richard Granata, Dr. Marvin White and Dr. John Fisher whose advices are very helpful on this project. And I also wish to thank Mr. Zuozhang Ma, Mr.Yi Zhou ,Mr. Malcolm Chen, Mr. Maurice Caldwell, and other people who have helped me on this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following operators used in the thesis are defined as in C language:

operator	function
&	bitwise and
|	bitwise or
~	bitwise complementary
^	bitwise exclusive or

ABSTRACT

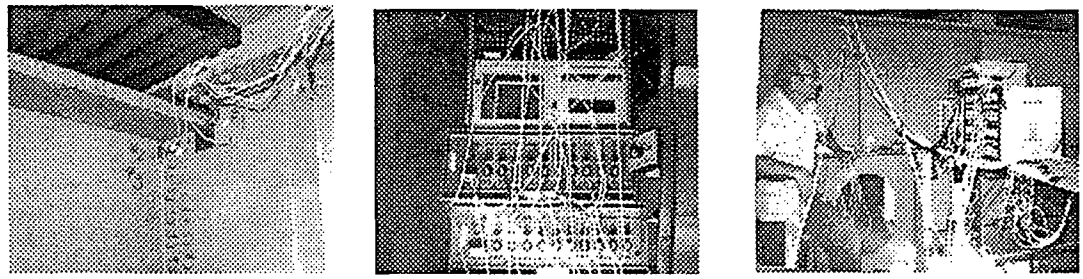
Design of a remotely accessible, highly reliable and economically affordable monitoring system for large structures is presented. The system can be used to monitor large structures such as bridges, railways, highways, ... Basic concepts about monitoring large structures and rainflow counting algorithm are reviewed. Demonstration of the system design is presented by a C program. Based on VLSI signal processing technology, design of the fatigue data processing chip is discussed in detail. Simulation of the chip design has been carried out using GDT tools and Magic tool.

CHAPTER 1 INTRODUCTION

Current studies in a specific field of science and technology are not only developing within itself but also growing up in other possible fields in today's world. This project is such a mixture. Using of VLSI (very large scale integrated circuit) signal processing technology for application of civil engineering, this project introduces a remotely accessible, highly reliable, and economically affordable monitoring system of large structures such as bridges, railways , etc... Many of these large structures in America or in other countries were built up tens of years ago. The endurance of these old structures has been becoming a remarkable problem today. The conditions and performances of these old structures should be properly examined for safety reasons. Repairing should be applied to damaged structures if necessary. Many methods are introduced to the inspection of large structures. In civil engineering, cumulative fatigue damage analysis procedures are usually employed to estimate fatigue life of large structures. A structure is considered safe during its fatigue life. When its fatigue life ends, the structure requires examination. This method allows engineers to relate the endurance of actual components to sample laboratory specimens. Fatigue life of the specimens are determined for constant amplitude tests. Real structures seldom, if ever , experience constant amplitude loading. Therefore some type of cycle counting scheme must be employed to reduce a complex irregular loading history into a series of constant amplitude events. The most accurate fatigue life estimates are obtained using an analysis based on the strain at the most highly stressed/strain location. Rain flow counting is an essential part of these procedures. And this system is also based on this counting method.

In current practice, civil engineers have to carry heavy equipment to the tested structures as shown in Fig 1.1. Attach sensors on the different significant locations of the structure. One recording tape will be filled up in two or three hours. They can repeat the same

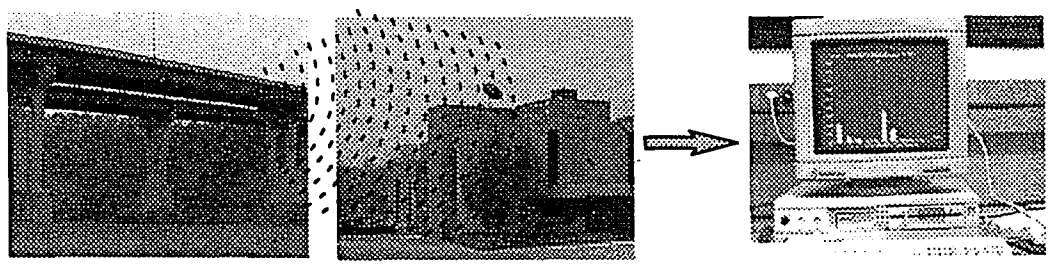
procedure in several days and then end an investigation of a structure. Above method can



sensor on structure equipment in van engineer working on tested structure

Fig 1.1 Current Practice

only give civil engineers a very rough estimation of fatigue life. They believe the precision of the result will be improved a lot if they can collect data in a long time (like one year). This project is to investigate the possibility of applying advanced signal processing technique and VLSI technology to achieving this goal. We hope to achieve a real–time data processing on the structures so that the monitoring time can be theoretically as long as the existence of the structure. A VLSI chip is designed to perform fatigue data processing in real time. It



Modules on structure antenna receiving signal computer in central control

Fig 1.2 Future System

means that the chip will process the data in real time and update the result – the histogram, and then discards the processed data and only keeps the result. By our design now, the chip can work continuously for at least one year without refreshment under normal traffic volume. And the chip and other necessary circuits can be integrated into a small module.

The ideal design of the whole system is sketched in Fig 1.2. The communications between the remote modules on tested structure and the central control can be carried out by radio signal in future system. Therefore engineers will avoid field work most of time and obtain more accurate result.

The application of the system will upgrade the studies of inspection of a structure and provide information to repair damaged structure and improve the design of the new structures.

CHAPTER 2 BASIC CONCEPTS OF MONITORING FATIGUE DAMAGE

2.1. ESTIMATION OF STRUCTURE FATIGUE LIFE

Many methods are involved in monitoring large structures. Estimation of remaining fatigue life plays an important role among the methods of inspection of a structure. Fatigue life is defined as the length of time before an initially microscopic flaw in a critical structural detail develops into a crack of certain length (typically about an inch or the thickness of the material) and becomes detectable by normal non–destructive inspection techniques (most commonly visual observation). Structures will perform safely until its remaining fatigue life becomes zero. After this point the structure may enter a dangerous period, an examination is required. Usually some repair work may be applied to these structures.

The principal parameters controlling the fatigue life include the nature of the structural detail, the stress range experienced by the detail, the volume and variation of live load traffic on the bridge. Dependent on the nature and fabrication of the detail, its fatigue characteristics are represented by the S – N Curve of the appropriate category as shown in Fig 2.1 . While the S–N curves are referenced to constant amplitude stress cycles , the stress cycles experienced by actually structural details vary over a considerable range on account of the variability of truck weights, truck configuration, and location of vehicles traversing the bridge. A method is needed to assess the fatigue damage under variable stress range spectrum. One such method uses an equivalent constant amplitude stress range, which causes the same damage and fatigue life as the actual variable stress range spectrum. The traffic volume information is needed to establish the number of stress cycles to convert the fatigue life into a length of time. In essence ,the fatigue evaluation of a bridge structure involves the establishment of the fatigue category of the critical detail, the estimation of an

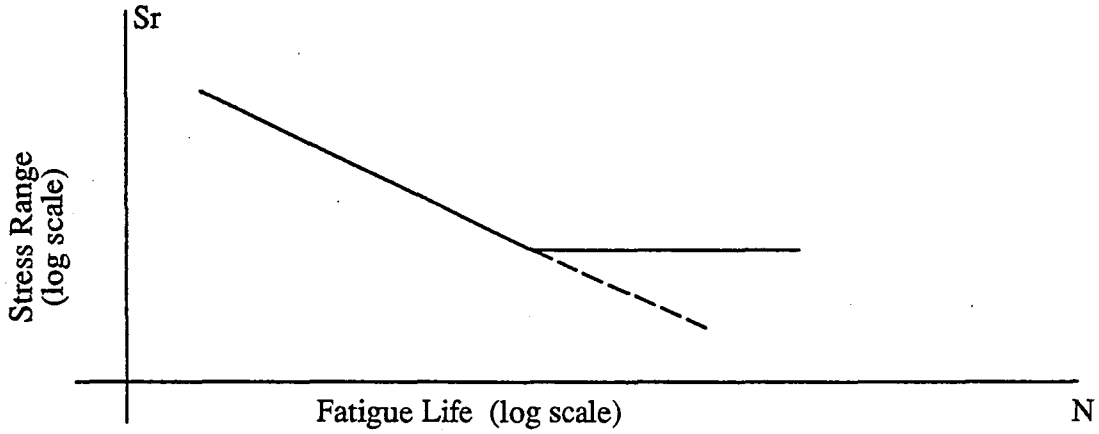


Fig 2.1 Typical fatigue strength curve

equivalent stress range, the estimation of the number of stress cycles experienced by the structural detail, and finally the estimation of the remaining fatigue life in terms of time units.

While the equivalent stress range is updating from histogram, the total cycle N_t (total fatigue life) which the bridge can afford under this certain equivalent stress range can be checked out from the S – N curve. The number of cycles to date N_p can be derived from the historical traffic information if it is available. Otherwise N_p can be obtained from a rough estimation. The degree of cumulated fatigue damage to date is estimated by comparing the cycles to date N_p with the total fatigue life N_t. The difference between these two values is the remaining fatigue life,N_r

$$N_r = N_t - N_p$$

Then we can convert N_r into a length of time during which the structure will perform normally. If $N_p > N_t$ the structure may enter a dangerous period. Some examinations and repair should be carried out.

2.2. MEASUREMENT OF STRESS RANGE

The best way to obtain the actual variable stress range spectrum requires the field measurement of stresses under actual traffic condition. Strain gauges are generally used at the location or detail of interest on the structure, and strains are monitored and recorded over a period of time sufficiently long to provide a realistic representation of the traffic pattern (now about two days which civil engineers consider may not be enough). This project is to design a small device which can process data on the monitored structure to generate the stress range spectrum – histogram. Monitoring time from this new method can be one year. The shape of the curve generally resembles that of an influence line, but with superimposed strain fluctuation reflecting the vibration of the structure due to the moving loads.

The strain–time record over a period is first reduced to a stress range histogram, showing that the frequency of occurrence of stress cycles with various range magnitudes. Several methods can be used to count the number of cycles. But the most common method selected by civil engineers today is a method called rain flow counting(see chapter 2 for detail) which has been used in this design too.

Fig 2.2 shows an example of measured stress–range histogram in which the ordinates show the fraction of stress cycles having range magnitudes within each finite interval. The equivalent constant amplitude stress range is commonly determined by invoking Miner's rule, which stipulates linear cumulation of fatigue damage, or

$$\sum n_i/N_i = 1$$

where n_i = number of stress cycles with range magnitude S_{ri} , and N_i = fatigue life corresponding to constant stress range S_{ri} as determined from the appropriate S – N curve. Since the typical S – N curve , for steel and concrete alike, has a gradient of –1/3 , the equivalent constant amplitude stress range based on Miner's rule is the "root–mean–cube'

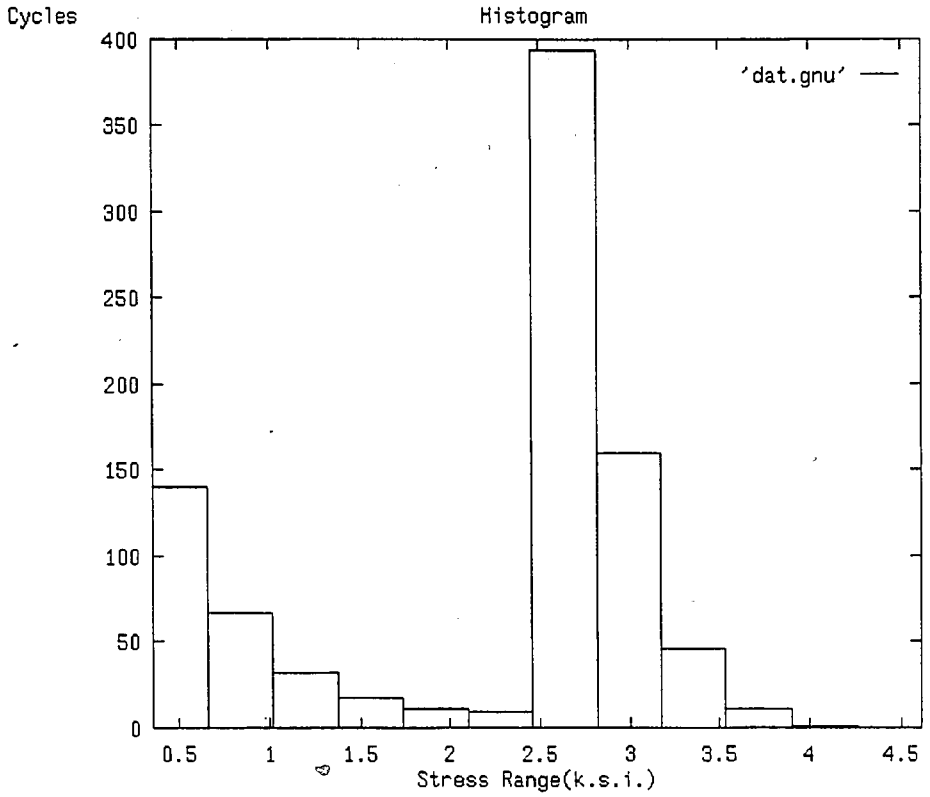


Fig 2.2 Histogram

value of the weighted sum of variable stress ranges cubed .

$$S_{re} = (\sum \gamma_i \, S_{ri}^3)^{1/3}$$

where , γ_i is fraction of occurrence of stress range S_{ri}, or $n_i / \Sigma n_i$. It is to be noted that because of the typically skewed distribution of the stress range histogram, the equivalent stress range S_{re} is significantly lower than the maximum stress range $S_{r,max}$.

2.3. RAIN FLOW COUNTING

Though there are many methods of cycle counting in general use, more and more engineers prefer rain flow counting method today. This method is also implemented in this system. The outstanding feature of the Rainflow Method is that it is carried out on the basis of the stress–strain behavior of the material being considered. The cycles which are

extracted are consistent with those in constant amplitude tests on which the life predictions are invariably based.

The result can be obtained using the analogy of rain running down a series of pagoda roofs. The strain–time record is drawn with the time axis drawn vertically downwards as shown in Fig 2.3. The general rules for rain flow counting are :

1. Rainflow begins at the beginning of the strain signal and at the inside of every peak.
2. Flow initiating at a maximum drips down until it comes opposite a maximum more positive than the one from which it started. Similarly, flow initiating at a minimum drips down until it comes opposite a minimum more negative than the minimum from which it started
3. Rain stops when it meets rain form the roof above.
4. The beginning of the sequence is a minimum if the initial straining is in tension.
5. The horizontal length of each rainflow is counted as half cycle at that stain range.

stain	counts
10	
9	
8	1
7	
6	1
5	
4	
3	
2	
1	2

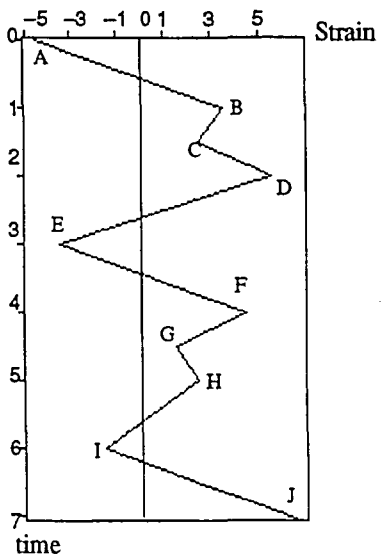


Fig 2.3 An example of rain flow counting

The table in Fig 2.2 shows the rain flow counting result of the strain signal. Chapter 4 introduces a subroutine of C program of rainflow counting algorithm and explains how the counting works on above signal.

CHAPTER 3 SYSTEM DESIGN AND APPLICATION

3.1. AN OVERVIEW OF THE MONITORING SYSTEM

The system will consist of a number of processing modules on the tested structure and a central device which controls these remote modules. The system block diagram is shown in Fig 3.1 as below.

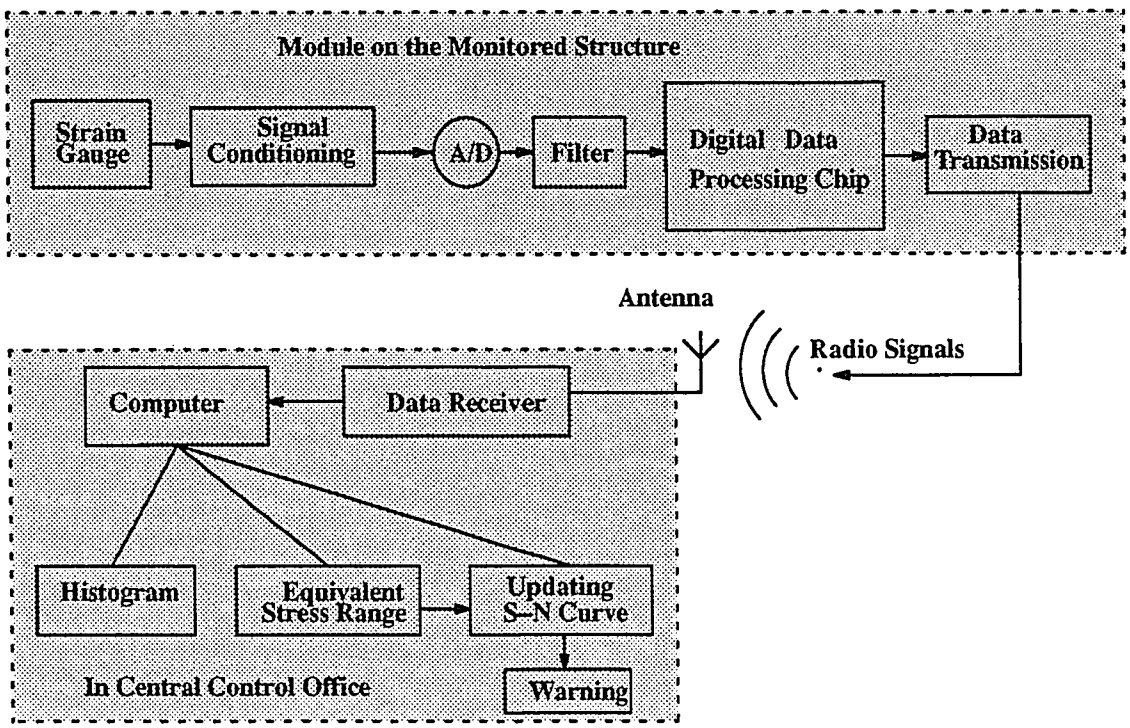


Fig 3.1 System block diagram

There is only one processing module shown in the above diagram, but actually there will be a number of modules attached to different significant locations of the inspected structure and remotely controlled only by one central office. The communication between modules

and the central control will be carried out by radio signals. Engineers can send out commands to remote modules and receive data from modules through a computer in the central control office. Because of application of large size memory in the module there is no need to refresh monitoring work within a short time, the module can support continuous monitoring at least one year under normal traffic volume. Thus, engineers will avoid field work in most of time. Thus, the system is able to achieve a time–unlimited monitoring.

3.2. DESCRIPTION OF THE PROCESSING MODULE

The upper shadowed part in Fig 3.1 shows the components of the module. Because of application of VLSI technology the module can be integrated into very small physical size. Strain gauge generates a voltage signal which represents the strain from the vibration of the structure. Signal conditioning will include a pre–amplifier circuit and a noise–cancelling circuit in order to eliminate all kinds of noises which are mainly high frequency interference and adjust the dynamic range of signal to an appropriate value. A/D converter converts analog signal to digital signal. Digital filter will filter out noise at very low frequency.10–20hz. Digital data processing chip does the rainflow counting and histogram generation. It processes data in real time and keeps on updating the histogram. It is the core of the system. And detail description of this part will be discussed in later chapters. In transmission part, we are planning to achieve it by radio signal. It can receive command or send back the histogram.

3.3. DESCRIPTION OF THE CENTRAL CONTROL

The lower part in Fig 3.1 shows , first all, the radio receiver will receive the data of histogram from the remote modules. The receiver could be a radio modem. Then computer will read in the data of histogram and generate desired results , such as updated histogram , S–N curve and warning signal. Computer also can generate control signal which will be sent

back to the remote module by radio modem. Therefore, a software which supports above jobs is required, and for design purpose we also need a program to simulate the design of the system.

3.4. APPLICATION OF THE SYSTEM

This program is written in C language, compiled using Borland C++ on PC. This program has a simple version running under unix in order to process large data. The propose of this program is to improve our understanding of related concepts from civil engineering , to verify the design since the system will perform the same task as the program does, and to help us and civil engineers to determine parameters used in the system.

We have tested the program by experimental data from laboratory, and real data from bridges and railways. Finally, our result from software program agrees with the result from hand calculation of our civil engineering colleagues. This provides the basis for verification of our design. A modified program will be used as the software supporting the future system. This program provides a menu interface. After the program starts, first two pages are title page and a brief introduction, then the main menu shown in Fig 3.2. A brief description of each function in main menu is given as follows:

1.System diagram

This page will present a system diagram as shown in Fig 3.1 , but the flashings in the diagram will indicate which part of system is working or what kind of current results is available and whether the monitored structure is in danger.

2.Set Parameters

Invoking this function follows a submenu shown in Fig 3.3.

1. Sensor location number indicates which module the central control connects to. It gives the location of the sensor on the monitored structure.
2. Number of used cycles (power of 10) is the number of cycles the structure have experienced from very beginning.

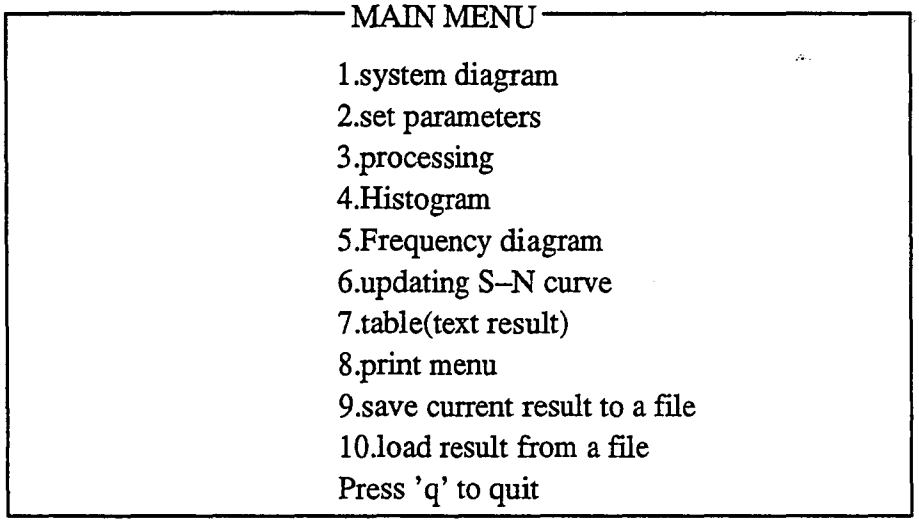


Fig 3.2 Main menu

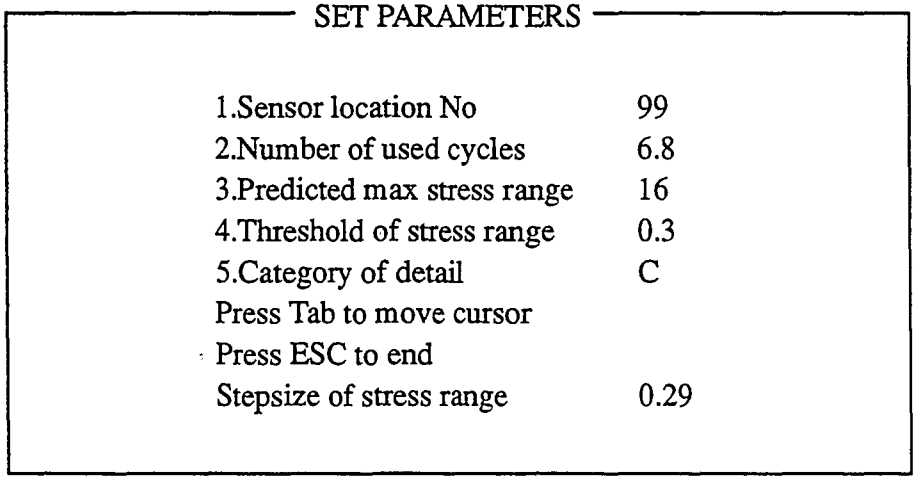


Fig 3.3 Parameters

3.Predicated Max Stress Range is the dynamics range of the signal from strain gauge.

4.Threshold of Stress Range makes the rain flow counting ignores cycles of small strain range.

5. Category of monitored detail indicates which S – N curve is applied to analysis.

6.The stepsize of stress range will be automatically shown up after item 3 is determined. Since the number of classes of strain range is 32, the stepsize of stress range will be

dependent on a calculation of the maximum strain range and the gain of the pre–amplifier.

3.Processing

In this part the program will processing the data from remote module, including data conversion, calculation of equivalent stress range and check it in S – N curve and turn on warning signal if necessary.

4. Histogram

This function displays the histogram as shown in Fig 2.2

Above histogram shows the results of an experiment in the ATLSS laboratory.

5. Frequency Diagram

This selection shows the same diagram as histogram except that the value of vertical axis is the percentage of cycles of each class in number of total cycles.

6. S –N Curve

This page shows the S – N curve of this detail and the equivalent stress range and fatigue life of the detail under this equivalent stress range.

7.Show Table

Show text result in a Table.

8.Print

This selection follows a sub menu :

1.print out the histogram (if a Laser printer is connected)

2.print out the S – N curve (if a Laser printer is connected)

3.save histogram as a postscript file

4.save S – N curve as a postscript file

5.print out the table of result(in item 7)

9.Save

Save current result to a file which can be loaded into the program again.

10.Load

Load result from a file generated by item 9

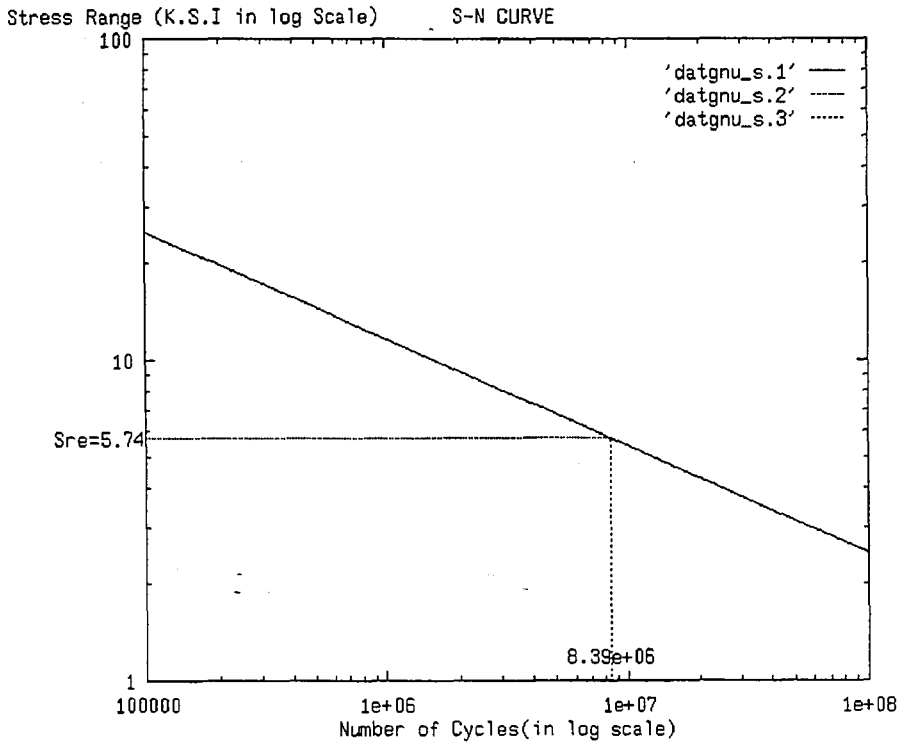


Fig 3.4 S – N curve

3.5. BRIEF EXPLANATION OF THE SOURCE CODE OF THE PROGRAM

This program is written in C language, complied using Borland C++ on PC. It has been tested on EGA and VGA monitor. The pure source code without comments has about 2,000 lines. Main parts of the program is written for interface. And GNU software 'gnuplot' is called into this program to support generating postscript file of graphics and printing on Laser printer. Many graphic functions and window functions of Borland C++ are used in this program. Chapter 4 introduces a subroutine in this program which performs rain flow counting as the algorithm described in Chapter 2. It is the soul of the program. One important part of circuit design is based on it.

CHAPTER 4 DESIGN OF THE FATIGUE DATA PROCESSING CHIP

4.1. OVERVIEW

The design of the chip is based on the fatigue life estimation algorithm, rain flow counting algorithm, and system application requirements described in chapter 2. The chip can be divided into four parts. Fig 4.1 shows the block diagram of the chip.

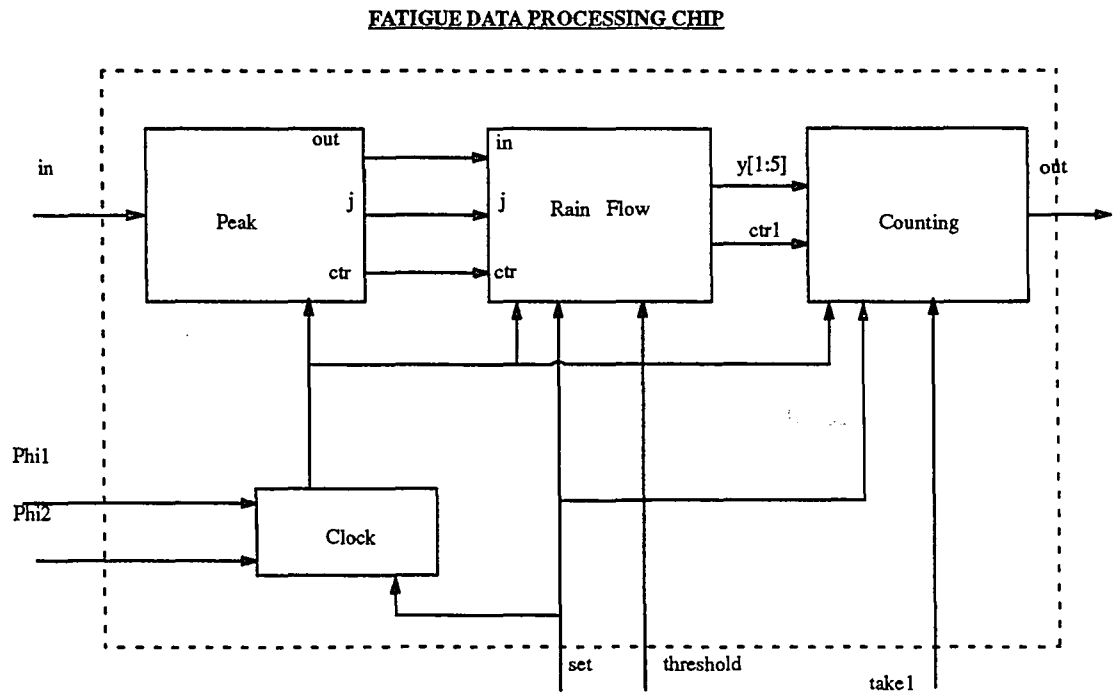


Fig 4.1 Block diagram of fatigue data processing chip

The function of the chip can be described in the following three steps.

1. Detecting the positive and negative peak values of input signal: block 'peak' in Fig 4.1 The block 'peak' is a digital peak detector which provides the peak values of the strain signal to the next circuit. It has one pin of serial input(the least significant bit first and most significant bit last). The output 'out', a 12–bits bus provides the peak values of strain to

rain flow part when the one bit output 'ctr' is high; one bit output 'j ' indicates whether 'out' is a maximum or a minimum(a positive peak or a negative peak).

2. Processing peak values to generate the value of stress range by rain flow counting algorithm : block 'rain flow' in Fig 4.1.

   Because the max stress range is divided to 32 classes in counting, y[1:5] (five highest bits of stress range) is sent to the counting part when output 'ctr1' is high. The 12 bits input 'threshold' is provided by a calculation of the gain of the pre–amplifier and the real threshold value of stress range.

3. Counting cycles, updating histogram and writing out histogram: block 'counting' in Fig4.1

   In block 'counting' there are 32 24–bits memory cells used to cumulating the histogram . When take1 is high, the circuit will be writing histogram out to one serial output pin without interrupting counting.

'set' in Fig 4.1 is the starting signal. The block 'clock' provides several clock signals with four different frequencies to the other three parts.

4.2. PEAK CIRCUIT

The function of this circuit is to find out the peak values (both maximum and minimum) of the input signal. Fig 4.2 shows the circuit. 'reg2' and 'reg3' are two chains of 12 single registers in serial connections. The circuit has a serial input 'in' and a parallel output 'out' and other two control signals 'ctr' and 'j'.

1. Comparator Design

The block 'comparator' in Fig 4.2 is a sequential comparator as shown in Fig 4.3. The comparator is designed based on a subtracter:

if in1 > in2 <=> in1 – in2 > 0, then the sign bit of full adder's sum will be zero,

out=0 oute=1;

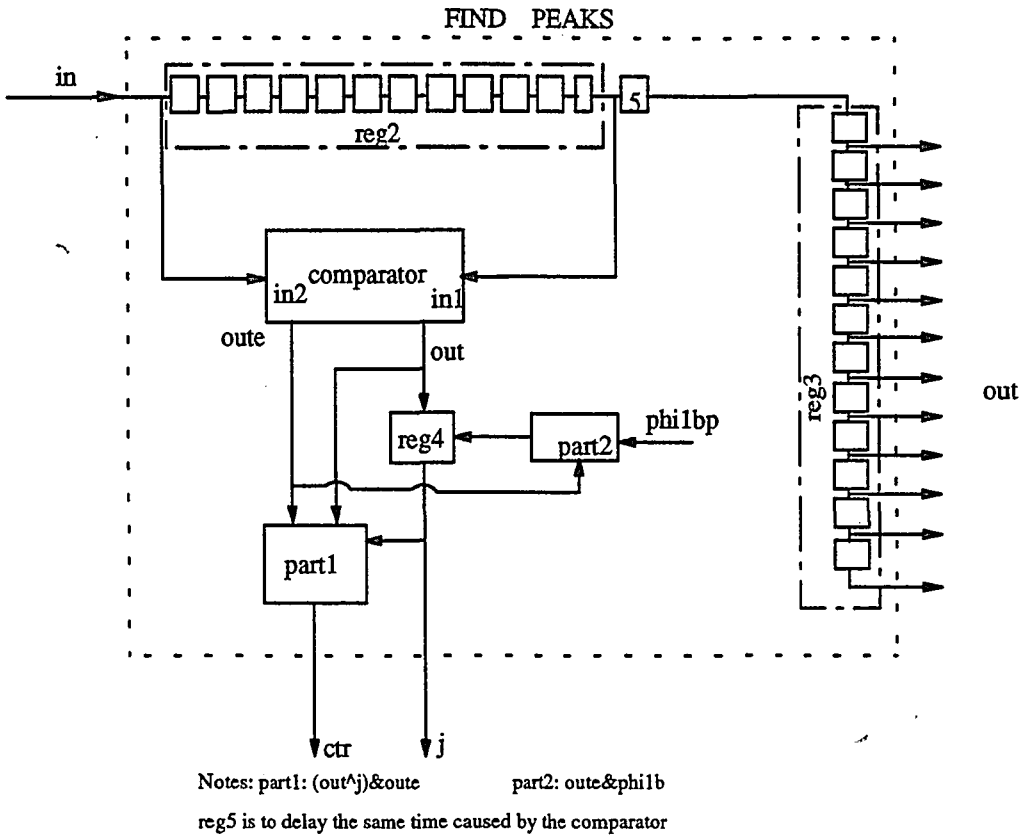


Fig 4.2 Peak

if in1 < in2 <=> in1 – in2 < 0, then the sign bit of full adder's sum will be 1.

out=1 ;

if in1 = in2 <=> in1 – in2 = 0, then all bits of full adder's sum at all cycles will be zero.

out=0 oute=0;

reg2 and part1 work together to determine oute=0 when the sum of full adder is all zero at every cycle or oute=1 when any bit of the sum is nonzero.

Two different frequency clock controls are used in the comparator, they are shown in Fig4.4

sample rate: Phi1b and Phi2b

bit rate: Phi1 and Phi2

When Phi1b is high, the least significant bit of new sample is read in.

2. 'Peak' circuit design

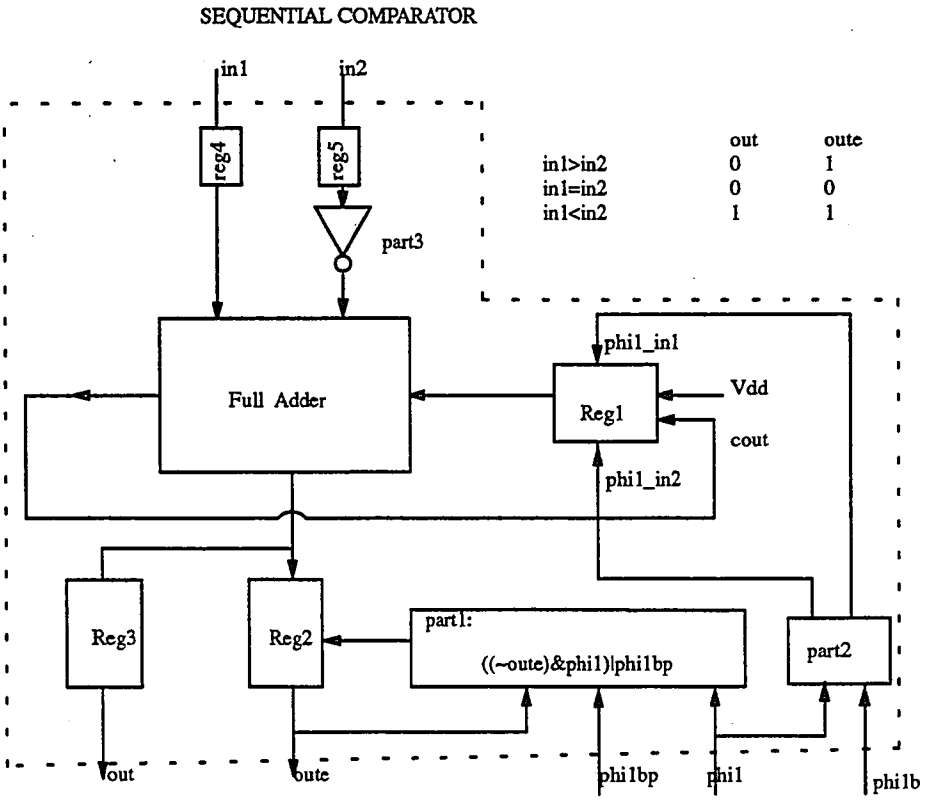


Fig 4.3 Sequential comparator

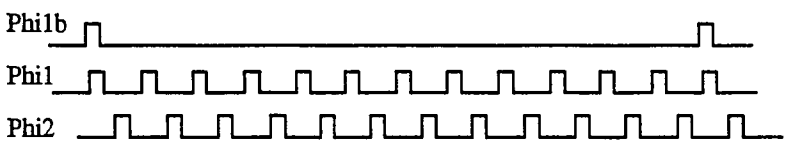


Fig 4.4 Phi1 Phi2 and Phi1b

Let's consider a typical wave signal as shown in Fig 4.5. First, we assign that:

case 1:

ctr = 1 and j =1 => 'out' is a maximum point as point 'c' in Fig 4.5.

case 2:

ctr = 1 and j =0 => 'out' is a minimum point as point 'i' in Fig 4.5.

case 3:

ctr = 0 => 'out' is neither maximum nor minimum.

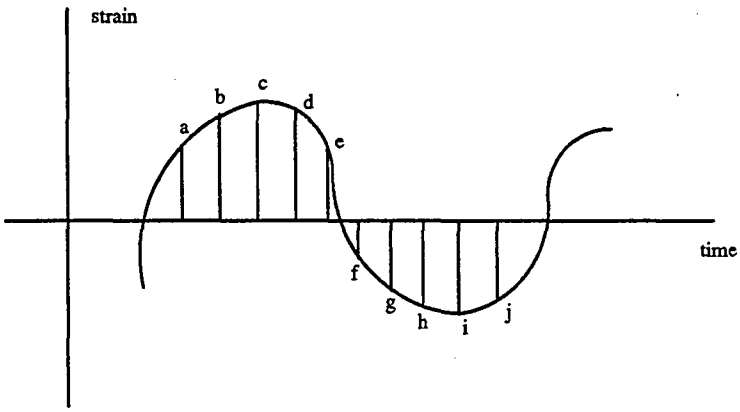


Fig 4.5 A sample of strain signal

Assume that the new coming point is b (in = b in Fig 4.2) , then in1 = a , in2= b in comparator, in this case j=1(to seek a max), the circuit will determine whether 'a' is a max.

Since a < b , i.e. in1 < in2 , then result of the comparator oute = 1, out = 1.

ctr = (out ^ j) & oute = (1 ^ 1) & 1 = 0.

It indicates that 'a' is not a peak value from case 3, and j in next cycle will be current 'out' = 1, unchanged, still to seek a maximum point. Similarly, b < c , then 'b' is not a peak value either. When 'd' is coming, c >d , i.e. in1 > in2, then result of the comparator oute = 1, out = 0, it follows:

ctr = (out ^ j) & oute = (0 ^ 1) & 1 = 1.

This indicates that 'c' is a peak value from case 1, and j in next cycle will be current 'out' = 0, changed, to seek a minimum point. Similarly, we can find that 'i' is a minimum point.

The following is a table of logical relations among ctr, j, oute, out and j in next cycle. From the table we can find that

ctr = (out ^ j) & oute

and the Phi1 of reg4 in Fig 4.2 is oute & Phi1b

Table of logic relations

out	j	oute	ctr	j in next cycle
1	1	1	0	1
1	0	1	1	1
0	1	1	1	0
0	–	0	0	unchanged

Table 4.1 Logic relations of peak circuit

4.3. RAIN FLOW

The design of this part requires understanding of rain flow counting algorithm in chapter 2.

1. C subroutine of rainflow algorithm

All the names of variables used in the program are the same as they are in circuit design and simulation.

```
1. rain(peak, record)          /*peak ,generated by upper routines is the peak */
2.                             values of signal of strain;*/
3.    float  peak;
4.    int    *record;                  /*record , an array of integer , stores the
numbers of      5.                          cycles in each class */
6.    {
7.          static float   n[3];   /*the three peak values of stress  */
8.          float   y;                    /* the value of stress from rainflow
algorithm*/
9.          char    s;                    /* the result of a comparison of n[2] and n[0]
10.         static char    p = 1;         /* parameter for memory refreshment */
11.       n[2] = peak;
```

```
12.     while (1) {
13.         if (j) {
14.             y = n[0] – n[1];
15.      s  = (n[0] > n[2])?1:0;
16.         } else {
17.             y = n[1] – n[0];
18.       s  = (n[2] > n[0])?1:0;
19.         }
20.             if (p&s) {                    /*  Q = p &s  in design */
21.      m[mp++] = n[0];                      /* 16 bits cp in design are decoded from mp */
22.      if (mp == 16)
23.                 p = 0;
24.             n[0] = n[1];
25.             n[1] = n[2];
26.             break;
27.         }
28.         count(y, record);
29.         if (mp == 0) {
30.             p = 1;
31.             n[0] = n[1];
32.             n[1] = n[2];
33.             break;
34.         }
35.         if (mp == 1) {
36.             n[0] = m[—mp];
37.             n[1] = n[2];
```

```
38.                 m[0] = 0;
39.                 break;
40.             } else {
41.                 n[1] = m[—mp];
42.                 m[mp] = 0;
43.                 n[0] = m[—mp];
44.                 m[mp] = 0;
45.             }
46.         }
47.}
```

The following is a brief description of some variables used in the above program.

Global variable 'j' in the above program has the same definition as it is in 'peak' circuit.

j = 1 , value of peak is a maximum point.

j = 0 , value of peak is a minimum point.

static char p is a flag for refreshment of counting when memory used in the circuit is full.

when p = 1, the counting is in normal case

when p = 0, the counting is in refreshment

Static float array n[3] stores wave form of peak values of the strain as shown in Fig 4.6.

There are four cases of n[0], n[1] and n[2](ignore the equal cases).

char s in program is the result of comparison of n[2] and n[0]

s = 1 in case (b) and (c)

s = 0 in case (a) and (d)

Line 13 to 19 in the subroutine calculate out value of s

Variable Q used in the circuit design is equal to p & s

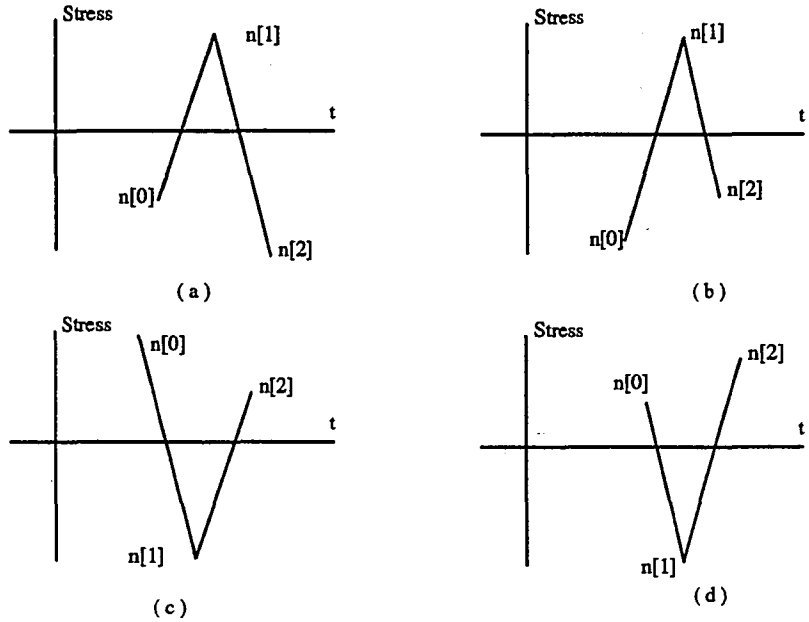


Fig 4.6 Four typical wave forms

Global float array m[16] is used as the memory to store peak values.

Global int mp is defined as the the index pointer of the m[16] array above.

Some explanation of the rainflow counting algorithm and the C subroutine as below:

In case (b) or (c) :

(i). rainflow counting algorithm:

No counting will be done now but the value of n[0] will move to memory, n[1] => n[0] , n[2] => n[1], and new peak value => n[2] in the next cycle.

(ii). In subroutine

assume that counting is in normal cycle, then

p =1

wave form is in case (b) and (c),then

s= 1

then p&s = 1,

the 'if' statement in line 20 is true, then m array reads in data of n[0] and increase memory point mp by 1

m[mp++] = n[0] ; in line 21

line 24 and 25 make shifts of data

line 26 'break;" quits the loop of statement 'while' in line 12.

return to main program. end this cycle and wait for the next peak value.

In case of (a) or (d) :

(i). rainflow counting algorithm:

stress range |n[0]–n[1]| is counted in, then the two points n[0] and n[1] are discarded, two new values are read back from memory (if available) as the new values of n[0] and n[1],and the new values of n[0], n[1] and n[2] will be checked out to see if they are still in the case (a) or (b). If so, then repeat this operation until they are in the case (c) or (b) or no data can be read back from the memory.

(ii). In subroutine:

p =1

wave form is in case (d) and (a), then

s= 0

then p&s = 0,

follows:

the 'if' statement in line 20 is false,

y = | n[0] – n[1] |; in line 13 – 19

stress range y passes to the subroutine count() in line 28. count() will count the stress range by its magnitude to 32 classes, and update the histogram.

line 29 –39 deal with the cases that memory has less than 2 data stored.

line 41 – 44 n[1] and n[0] read back the two data from memory and decrease memory pointer mp by 2.

then return to 'while' loop in line 12, repeats above procedures again till the subroutine runs to encounter the three 'break' statements:

(1) line 26, the current wave form is in case of (b) or (c)

(2) line 33, the current wave form is in case of (a) or (d), but memory pointer mp = 0 , it means that no data are stored in memory now.

(3) line 39, the current wave form is in case of (a) or (d), but memory pointer mp = 1 , it means that only one data point is stored in memory now, which can be read back.

In above three cases the subroutine needs to return and wait for a new peak value.

2. An example of rainflow counting

Let's consider rain flow counting of the strain signal shown in Fig 4.7 cycle by cycle.

stain	counts
10	
9	
8	1
7	
6	1
5	
4	
3	
2	
1	2

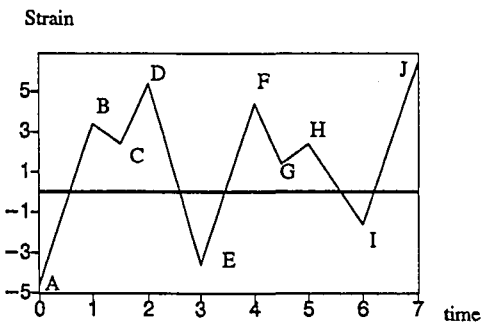


Fig 4.7 An example of rain flow counting

The followings are explanations of in each cycles:

Cycle 1:

wave form of A B C is in case of (b)

mp = 0, A => m[mp++]

m[0] = A

Cycle 2:

mp = 1,

wave form of B C D is in case of (d).

the stress range |B – C| will be passed to subroutine count(), $B - C \simeq 1.0$ as shown in Fig4.7. count() will increase the value in the class of stress range 1.0 by 1.

discard point B and C.

Cycle 3 :

since only one data A in m[0] can be read back to n[0], and mp is deduced to zero, then

n[0] = A n[1] = D

wave form of A D E is in case of (b):

the same as cycle 1 except:

Cycle 4:

mp = 1

wave form of D E F is in case of (c):

D => m[1]

Cycle 5

mp = 2

wave form of E F G is in case of (b):

E => m[2]

Cycle 6:

wave form of F G H is in case of (c):

mp = 3

E => m[3]

Cycle 7:

mp = 4

wave form of G H I is in case of (a):

the stress range |G – H| will be passed to subroutine count(), $G - H \simeq 1.0$ as shown in Fig4.7. count() will increase the value in the class of stress range 1.0 by 1.

discards point G and H.

then two latest data of memory will be read back

F in m[3] => n[1], E in m[2] => n[0] mp = 4 –2 = 2

Let's consider the wave form E F I, it is still in case (b)

E => m[2]

Cycle 8:

mp = 3

wave form F I J is in case of (d)

the stress range |F – I | will be passed to subroutine count(), $F - I \simeq 6.0$ as shown in Fig4.7. count() will increase the value in the class of stress range 4.0 by 1.

discards point F and I.

then two latest data of memory will be read back

E in m[2] => n[1], D in m[1] => n[0] mp = 3 –2 = 1

Let's consider the wave form E D J, it is still in case (d)

the stress range |E – D | will be passed to subroutine count(), $E - D \simeq 8.0$ as shown in Fig4.7. count() will increase the value in the class of stress range 6.0 by 1.

discards point E and D.

in next cycle : since mp = 1, only one data A in m[0] can be read back to n[0], and mp is deduced to zero. n[0] = A n[1] = J

Therefore , the rain flow counting result of the signal in Fig 4.7 will be the table in Fig 4.7.

3. Refreshment of memory

As shown in above example, the data A and J are still waiting for count. Refreshment of the counting is needed, the parameter 'p' in program is the flag for refreshment. When the memory is full, p will be set to '0' at line 23, then Q = (s&p) will be '0', the subroutine count() is called every time until line 30(p = 1) is executed when mp has been reduced to

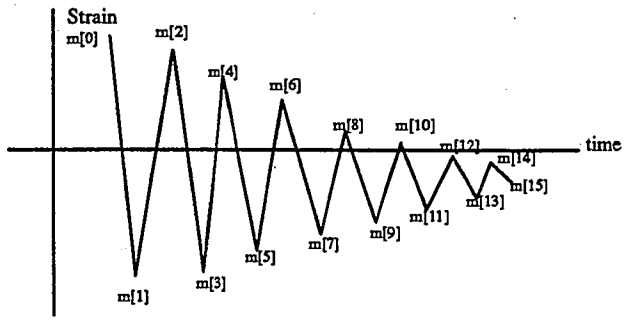


Fig 4.8 An example of full memory

zero, i.e. the all the data in memory have been taken back to count. The maximum number of data points to be stored in memory is 16 as shown line 22. Fig 4.8 shows an example of full memory. Memory refreshment has to be fulfilled in one period of input data (the period of peak value). It means that refreshment in one data period must include 1 counting for the current wave form and 8 counts for 16 data stored in memory, thus, the counting rate must be 9 times data rate. The clock variables of sample rate are still Phi1b and Phi2b, and clock variables of counting rate are noted by Phi1s and Phi2s here as shown in Fig 4.9

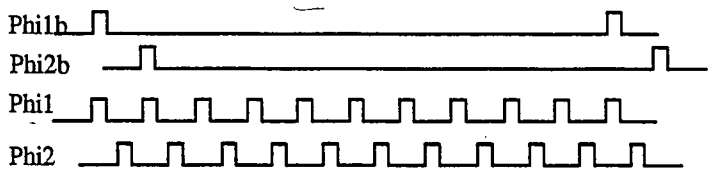


Fig 4.9 Phi1s, Phi2s, Phi1b and Phi2b

The following design of the circuit is based on above analysis and all the name of the parameters used in circuit are the same as they are in the program.

4.Circuit design of rainflow counting

The circuit block diagram of rain flow is shown in Fig 4.10. Boxes denoted by 1,2,3...7 are registers.

1. Some components used in the circuit

   a. two–input register

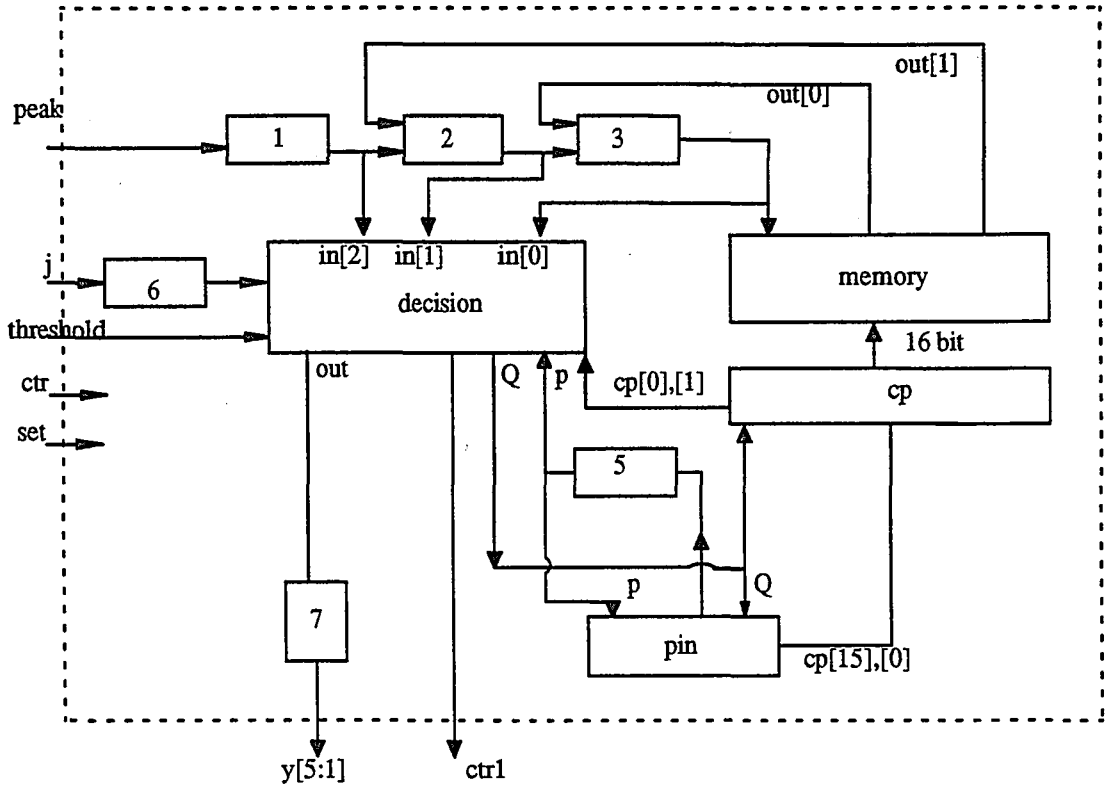


Fig 4.10 Rainflow counting

Fig 4.11 shows that two inputs in1 and in2 are controlled respectively by Phi1_in1 and

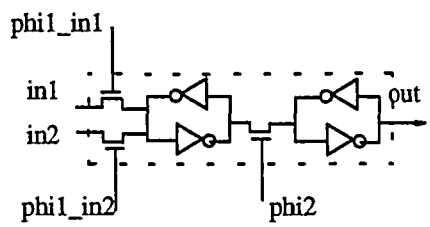


Fig 4.11 A two–input register

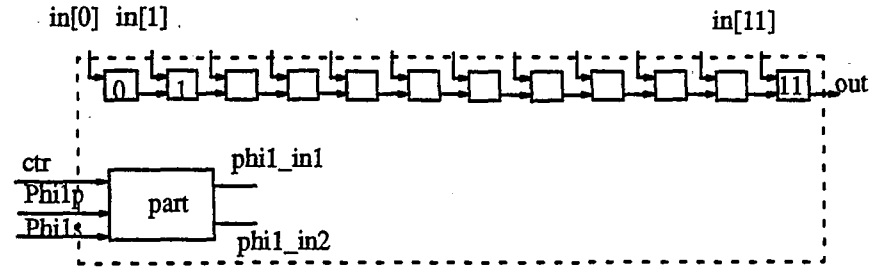
Phi1_in2.

when Phi1_in1 is high, the register will read in data of 'in1'

when Phi1_in2 is high, the register will read in data of 'in2'

b. parallel to serial converter

Fig 4.12 shows that Phi1p is for the parallel clock, Phi1s is for the serial clock and boxes denoted by 0,1,2,3...11 are two–input registers.



part:

phi1_in1=ctr&Phi1p
phi1_in2=ctr&Phi1s&(~Phi1p)

Fig 4.12 Parallel to serial converter

The function of 'part': Phi1_in1 = ctr &Phi1p

$$Phi1\_in2 = ctr \;\&\; Phi1s \;\&(\sim Phi1p)$$

Phi1s and Phi1p can be represented by Phi1ss and Phi1s in Fig 4.13 respectively.

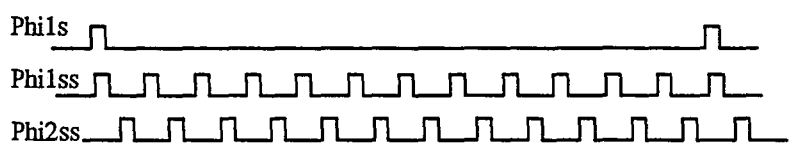


Fig 4.13 Phi1ss Phi2ss and Phi1s

when Phi1_in1 is high, input in[0], in[1] ... in[11] will be read in.

when Phi1_in2 is high, in[0], in[1] ... in[11] will be shift out to 'out' one by one.

c. qualified serial subtracter

Fig 4.14 shows that the function of circuit:

when j = 1, out = in1 –in2

when j = 0, out = in2 –in1

Reg1 is a two–input register which

$$Phi1\_in1 = Phi1ss(\sim Phi1s)$$

$$Phi1\_in2 = Phi1s$$

Phi1s is the sample rate and Phi1ss is the bit rate as shown in Fig 4.13. When Phi1s is high, in2 will be read as 1.

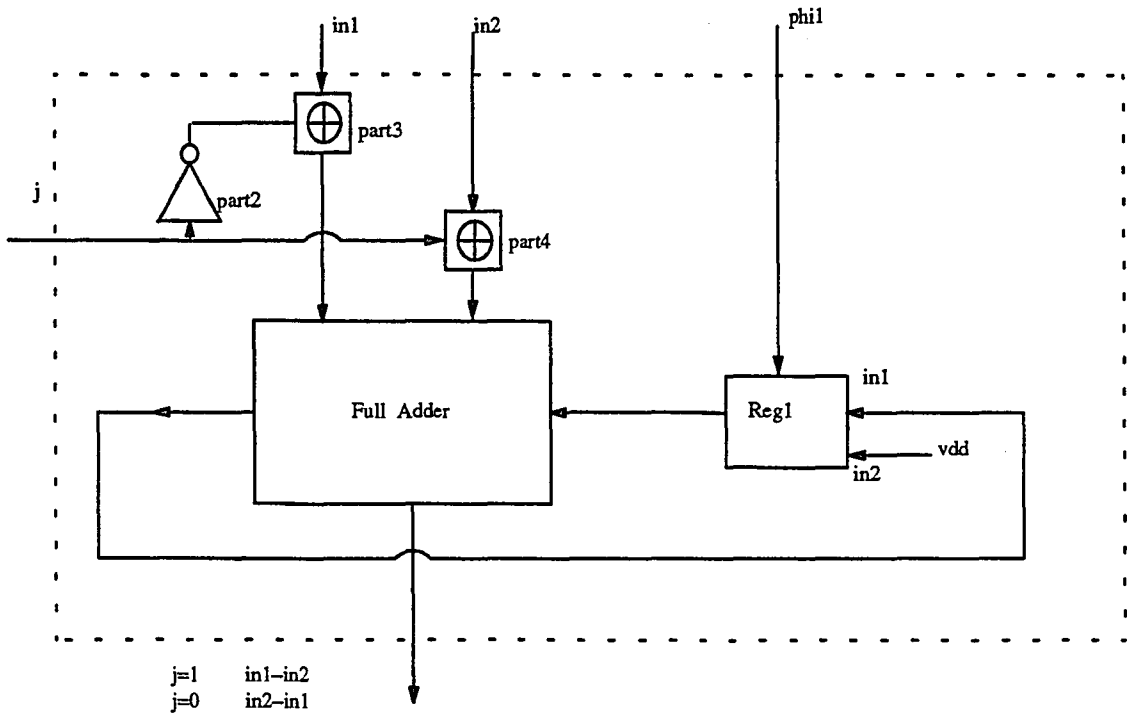


Fig 4.14 A qualified serial subtracter

Part3 and Part 4 are exclusive or gates. part2 is an inverter.

This circuit can be used to compute an absolute value of the difference of two numbers' subtraction or make a comparison.

d. register 1 in Fig 4.10

It is an array of 12 registers in parallel which is qualified by 'ctr' from peak circuit.

e. register 2 and 3 in Fig 4.10

They are two arrays of 12 two–input registers in parallel which should be qualified properly. These controls are related to deep comprehension of the rainflow counting algorithm, and needs four circuits to control them.

f. threshold circuit

The circuit in Fig 4.15 is very similar to circuit of parallel to serial converter except the connection between register 0 and 11. This connection keeps the data in registers

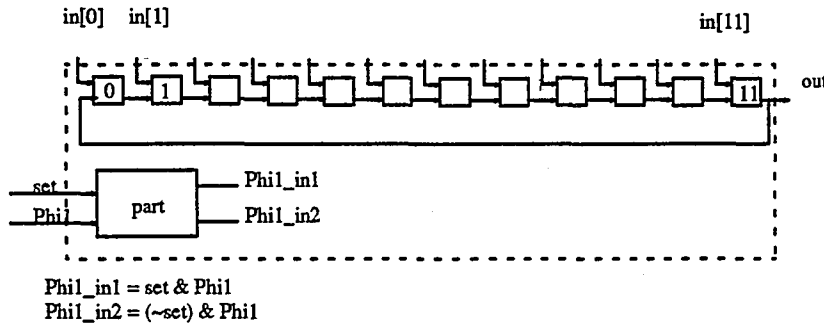


Fig 4.15 Threshold input circuit

shifting when 'set' is low. This make the circuit give out a constant serial output. The constant will be read in when 'set' is high.

g. block 'cp' in Fig 4.10

The circuit shown in Fig 4.16 is the decoding circuit of mp, output 'cp' is a 16–bits

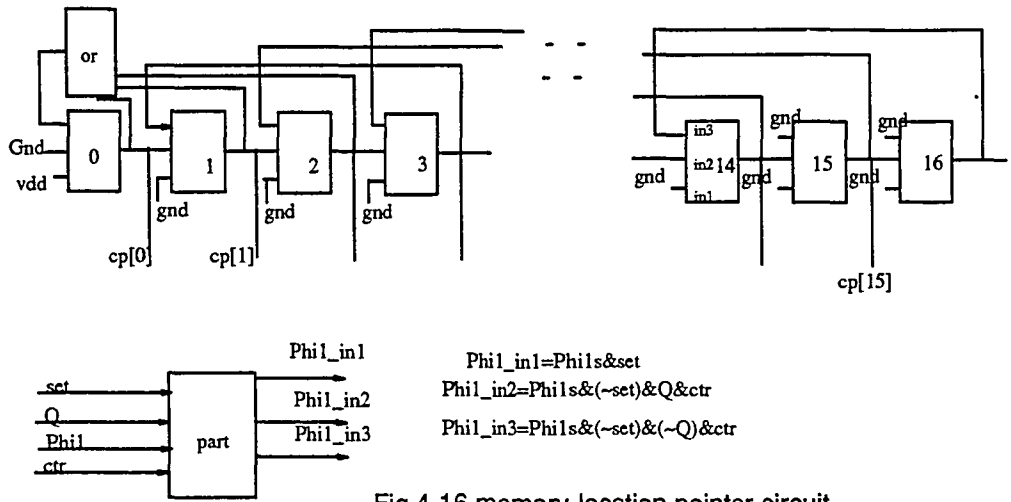


Fig 4.16 memory location pointer circuit

bus, which indicates the memory location. And it will perform the function of 'mp' in the C subroutine. When mp = i, cp[i] is high while cp[j] = 0 if j is not equal to i. Boxes denoted by 0,1,2 ... 16 are three–input registers (similar to two–input registers described in item a and three inputs in each register are noted as the same in register 14 The three clock variables:

Phi1_in1 = Phi1s & set

Phi1_in2 = Phi1s & (~set) & ctr

Phi1_in3 = Phi1s & (~set) & (~Q) & ctr

when set is high, all the registers initially read in 0 except register 0 reads in 1.

when Q = 1(meaning that the mp will be increased by 1 in next cycle), Phi_in2 executes. then the data will be moved forwards.

when Q = 0 (meaning that the mp will be decreased by 2 in next cycle), Phi1_in3 executes. the data will be moved backwards by 2 registers in normal case except when mp = 0 or mp = 1. The or gate will deal these two special cases.

h. memory in Fig 4.10

Fig 4.17 shows the memory inputs and outputs. Fig 4.18 shows the details of the memory. The memory cells shown in Fig 4.18 will repeat 3 times in horizontal direction and 8 times in vertical direction.

a = ctr & cp[i] & Q &Phi1

b = ctr & cp[i] & (~Q) &Phi2

c will connected to b in the lower array as the dashed line shown in Fig 4.16, this will make the memory write out two latest outputs at one cycle.

when Q = high , 'in' will be read in memory.

when Q = low, 'out0' and 'out1' will write out.

i. decision circuit in Fig 4.10

Fig 4.19 shows this part. Qsub1 and Qsub2 are the qualified subtracters. PtoS[ ] are the parallel to serial converters. Qsub1 will generate the value |n[0] –n[1]|, Qsub2 will give the result of s. |n[0] –n[1]| will be compared with threshold by comparator,

If |n[0] –n[1]| > threshold, the output T = 1, otherwise T = 0.

Then make the decision by

Q = p & s

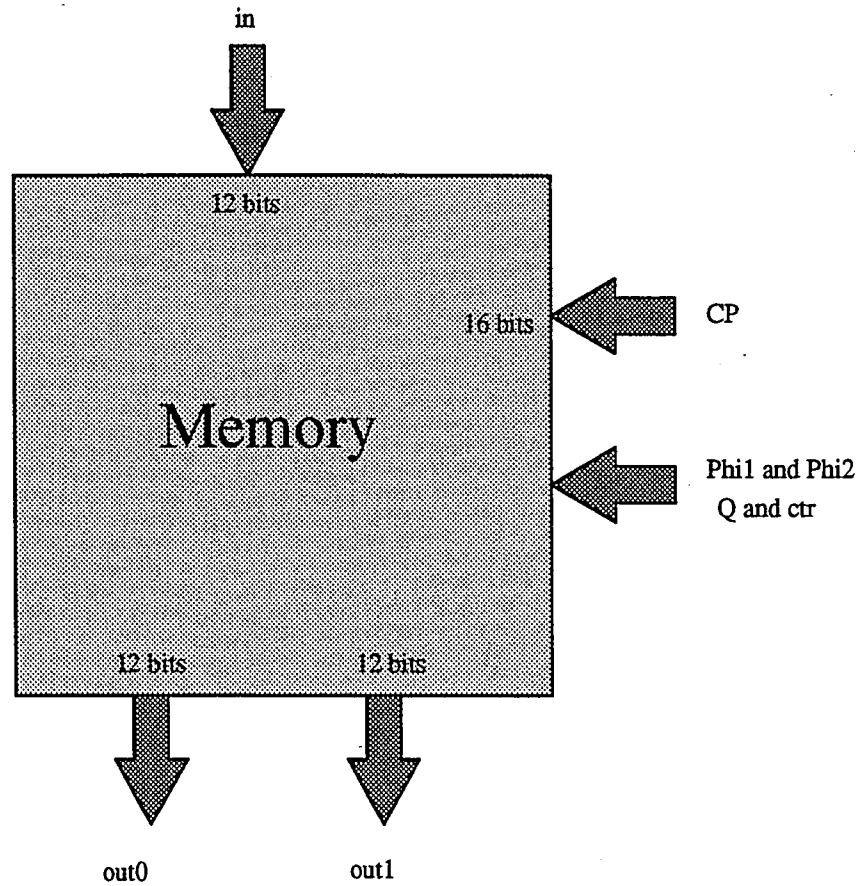


Fig 4.17 Memory

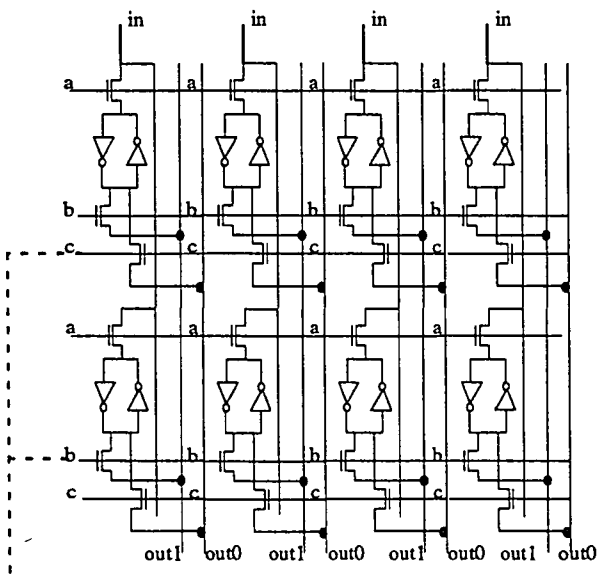


Fig 4.18 Memory cells

ctr1 =(~Q) & T &((Phi1s&(~Phi1b)&(~cp[0]))|(Phi1b&ctr))

'Q' and 'ctr1' control the rest circuit to work properly. Since Qsub1 , Qsub2, and

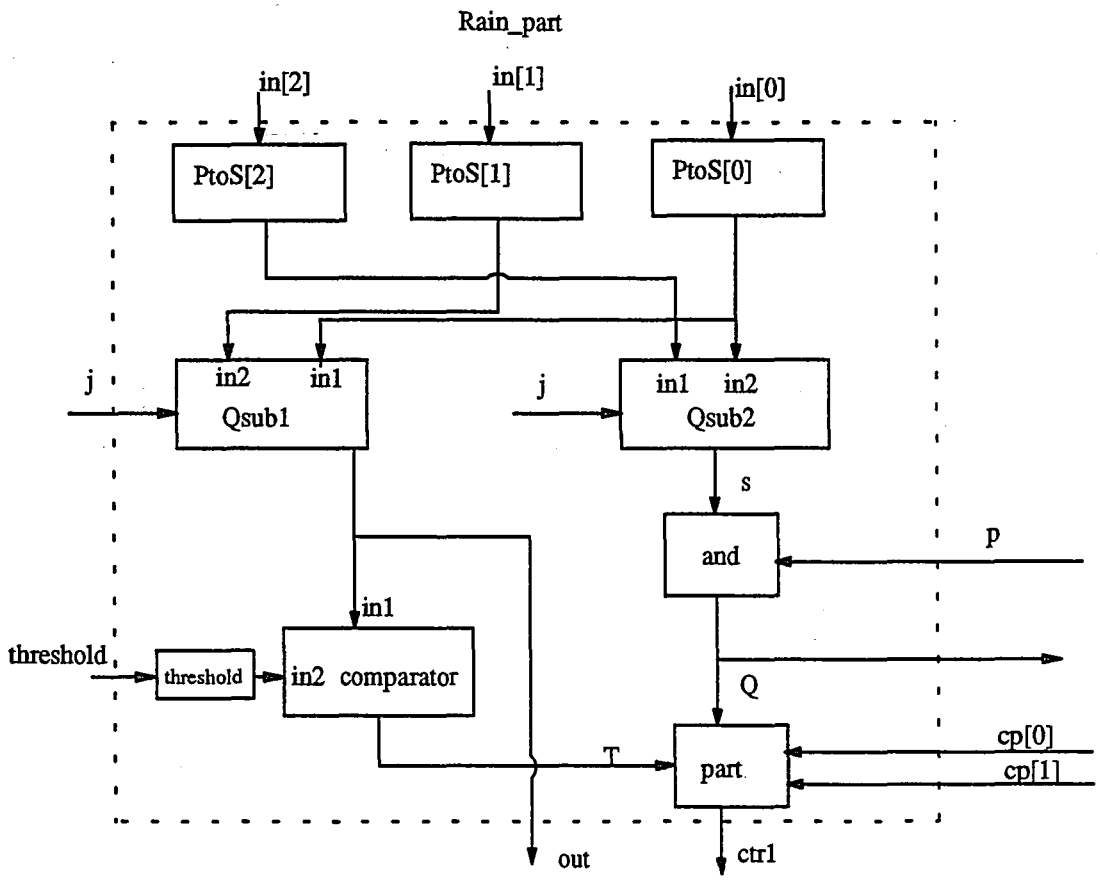


Fig 4.19 Decision circuit

comparator are designed to process serial data, another clock Phi1ss which is the bit rate for Qsub1, Qsub2, and comparator has the frequency 12 times of Phi1s as shown in Fig 4.13.

j. pin in Fig 4.10

It will generate refresh signal for next cycle,

pin = (p & (~cp[16]))| ((~p)&p[0])|set

It is very clear to see that

when cp[16] =1, the 16 memory cells are full. Then p = 0 to begin the refresh in next cycle.

when cp[0] = 1, and p = 0, all data in memory has been read back to count. then p = 1 in next cycle, turn it to normal cycle.

k. register 7

It is an array of 5 registers in serial connections used to keep the 5 highest bits(without sign bit) of stress range for the counting circuit later.

2. Explanation of circuit

As shown in Fig 4.9, register 1, 2, 3 will hold the data n[2], n[1] and n[0] respectively. Decision circuit in Fig 4.9 will determine whether the |n[0] –n[1]| is stress range(greater than threshold), if it is true, ctrl will be executed, otherwise ctrl keeps low. CP will work properly to meet the counting needs as above item g describe. register 1 and 2 must be controlled properly to read in data from the peak circuit or from memory.

4.4. COUNTING CIRCUIT

The function of counting circuit is to record the number of cycles under each class of stress range. Fig 4.20 shows the block diagram of the circuit.

1. Decoder

It consists of 32 5–input and–gates and 5 inverters. The function is to decode the 5 bit 'in' input to 32 bit output 'cp' as shown in Fig 4.20.

2. Control

The functions of the circuit are simple logic operations as below:

Phi1_in1 = set & Phi1ss

Phi1_in2 = Phi1ss & (~set)

in1 = Phi1s & ctrl

in2 = Phi1ss & (~ Phi1s)

in3 = Phi1s & (~ ctrl)

3. Memory cell

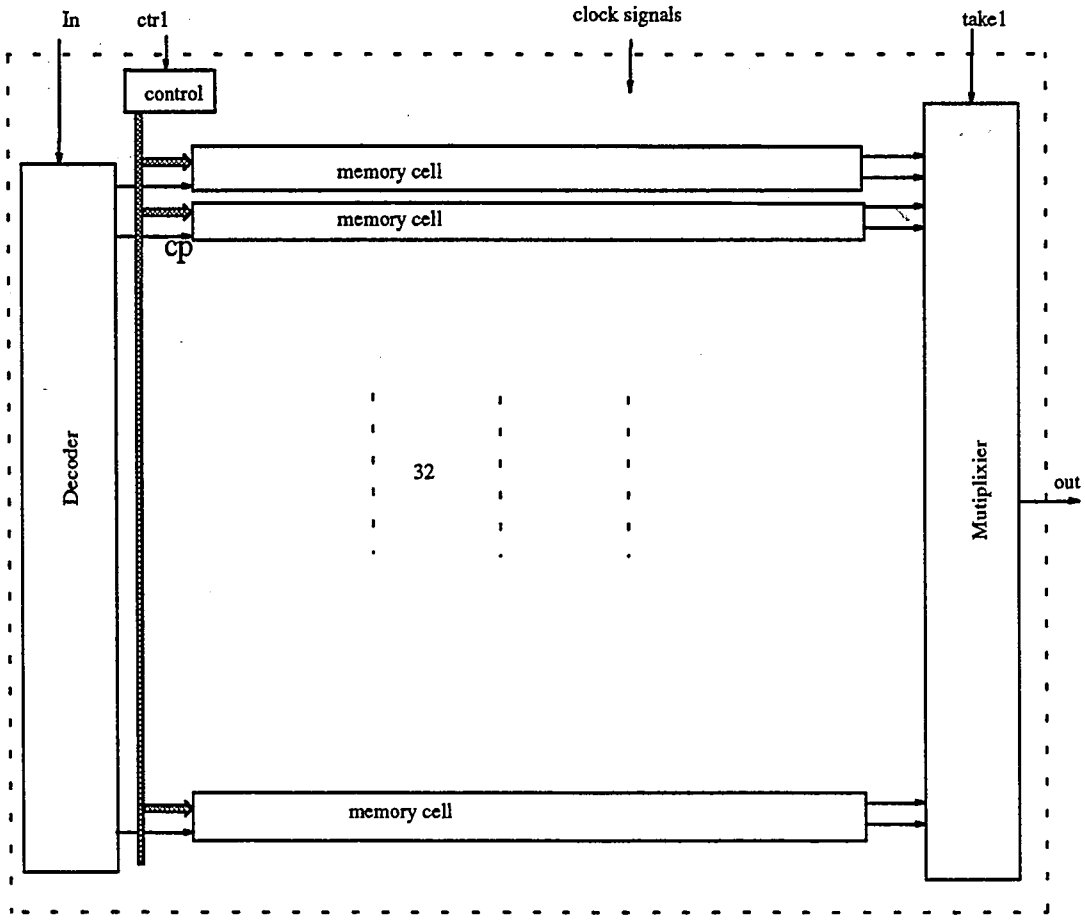


Fig 4.20 Block diagram of counting circuit

There are 24 two–input registers denoted by 0,1,2...23 as shown in Fig 4.21; regist is a three–input register. Inputs 'Phi1_in1 , Phi1_in2, in1, in2, in3' are read from above control circuit, Phi1_in1 and Phi1_in2 control the 24 two–input registers while the output of in1 & cp, in2 and in3 will control the regist as shown in Fig 4.21.

when 'set' = 1, all registers reads in zero, initialize all data to zero

when 'cp' is high, in1(Vdd) is read into regist at the cycle least significant bit is read in register 0, data in register ring 0 – 23 will be increased by one.

when cp is low, in2(GND) will read into regist data at bit cycle the least significant bit is read in register 0, data will not be changed.

output 'out0' and 'out1' will give out the result in every cycle.

4. Multiplexer

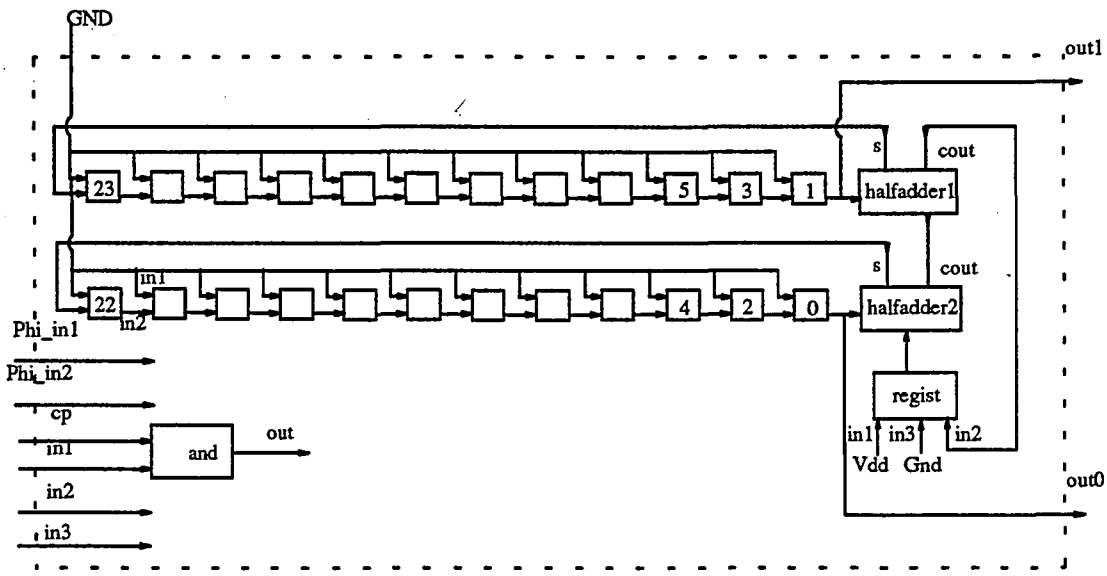


Fig 4.21 Memory cell

Fig 4.22 shows the multiplexer. which multiplexes the outputs of 'out0' and 'out1' from each memory cell to a serial output.

The boxes denoted by 0 1 .. 31 in Fig 4.22 represent 32 two input registers in serial connections.

when take1 is high, Phi1_in1 is executed. The data in registers have been initialized, register 0 is 1, all others are zeros.

when take1 is low, Phi1_in2 is executed. With the shift of the data in registers, the data stored in each counter_memory cells will be written out to the two inputs in1 and in2 of the two–input register shown in the right side of Fig 4.22. The frequency of Phi1m_in1 and Phi1m_in2 will be the same as bit rate of Phi1ss, but Phi2m is 2 times of bit rate of Phi1ss as shown in Fig4.23. It is used as a multiplexer. Finally, the circuit has one serial output

COUNTER_MUX

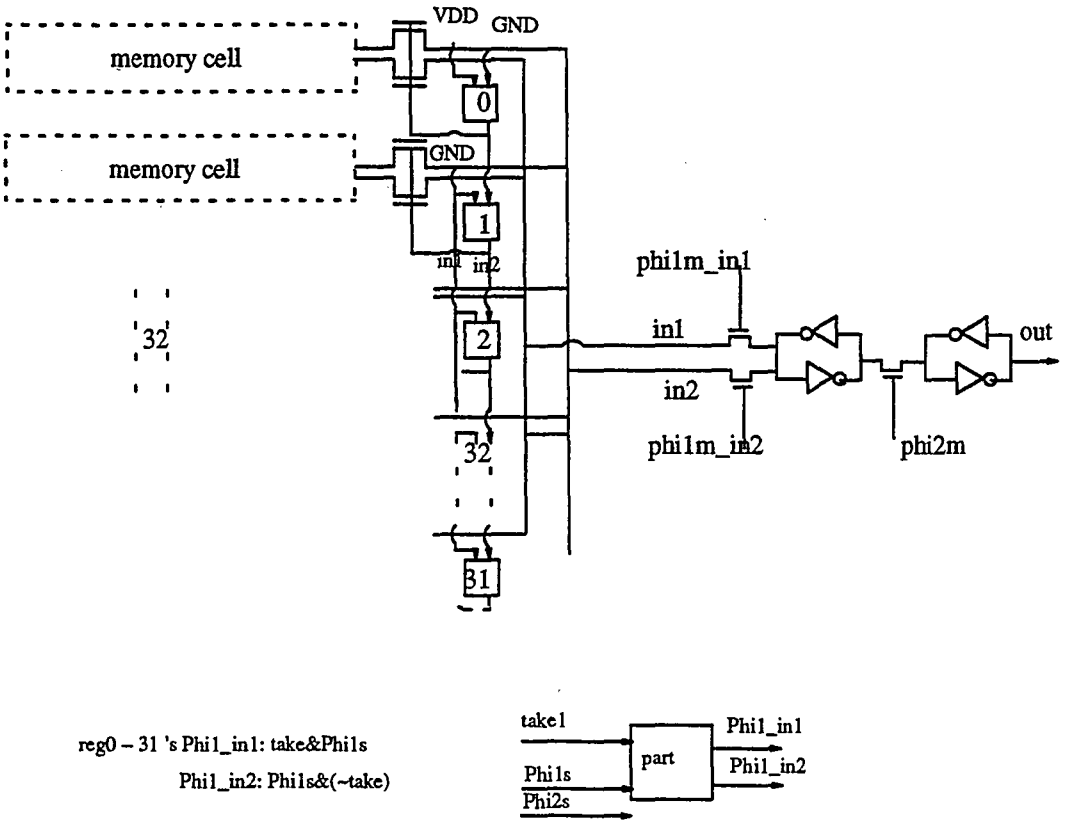


Fig 4.22 Multiplexer

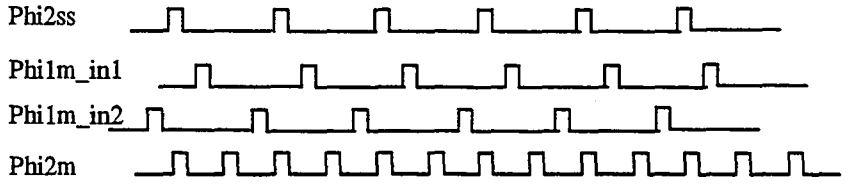


Fig 4.23 Phi1m_in1 Phi1m_in2 and Phi1m

4.5. CLOCK

1. A review of all clock signals used in above circuits

The fastest clock rate is the rate for Phi2m used in the circuit.

Assume the clock period of Phi2m is T_{Phi2m},

From Fig 4.4, Fig 4.8, Fig 2.2, we can obtain

$T_{Phi2ss} = 2\ T_{Phi2m}$

$T_{Phi2s} = 12\ T_{Phi2ss}$

$T_{Phi2b} = 9\ T_{Phi2s}$

$T_{Phi2b} = 12\ T_{Phi2}$

Therefore, if T_{Phi2m} is 1 unit, then

$T_{Phi2m} = 1$

$T_{Phi2ss} = 2$

$T_{Phi2s} = 24$

$T_{Phi2b} = 216$

$T_{Phi2} = 18$

2. Design of clocks

Because there is only one clock input for real VLSI circuit usually, there must be several sub–clocks designed to generate all clock controls the chip needs as shown in Fig 4.24. The following shows the design of clock2 in Fig 4.24.

3. An example of clock design

Fig 4.25 shows that the Phi1s and Phi2s are generated by Phi1ss and Phi2ss. Boxes denoted by 0,1,2... 11 are two input registers controlled by Phi1ss and Phi2ss. In initial cycle, these terminals connecting to Vdd or GND read in data array of

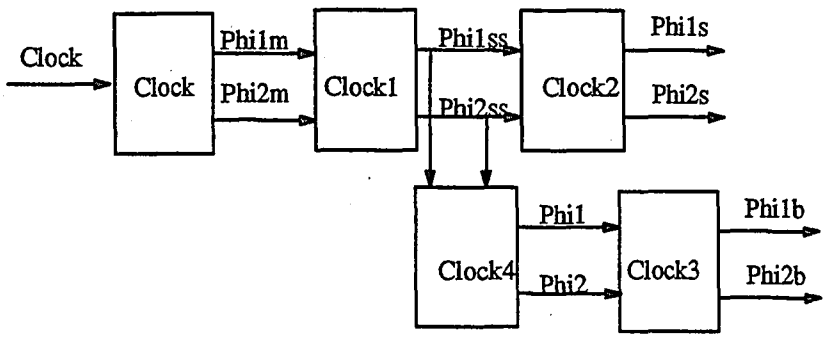


Fig 4.24 Clocks

SUB CLOCK

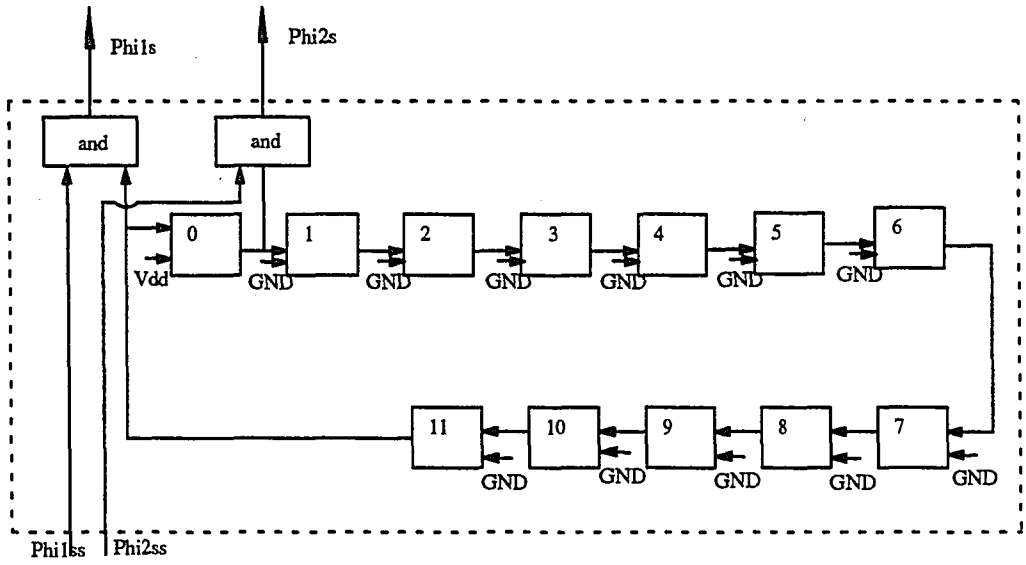


Fig 4.25 Sub clock

1,0,0,0,0,0,0,0,0,0,0,0 to register 0,1,2,..,11. The other clocks are designed based on the same idea.

CHAPTER 5 SIMULATION OF THE DESIGN

5.1. SIMULATION AT BEHAVIORAL LEVEL

As soon as we have the circuit design shown in Chapter 4, we are going to simulate the design by CAD(computer aid design) tool. The simulation consists of three steps: behavioral level, circuit level, and layout level. GDT tools are used at behavioral Level and circuit Level. In GDT tools, M language is used to write a behavioral model of design, Lsim is a simulator for models written by M language.

1.M modeling language

M language enables us to describe the design of digital and analog circuit for the Lsim simulator. M provides constructs for behavioral and structural modeling, as well as more traditional software programming, especially C language.

2.Two typical M instances

The following is a M model for a parity checker:

```
MODULE parity_checking()
{
    IN LOGIC in1;
    IN LOGIC in2;
    IN LOGIC in3;
    OUT LOGIC out;
    INITIALIZE{
        out=UNKNOWN;
        }
    SIMULATE{
```

```
    out=in1^in2^in3;
      }
}
```

As we can see above, most of the operators used in M language are the same as C language.

The followings is the simulation result of this parity checker from the Lsim:

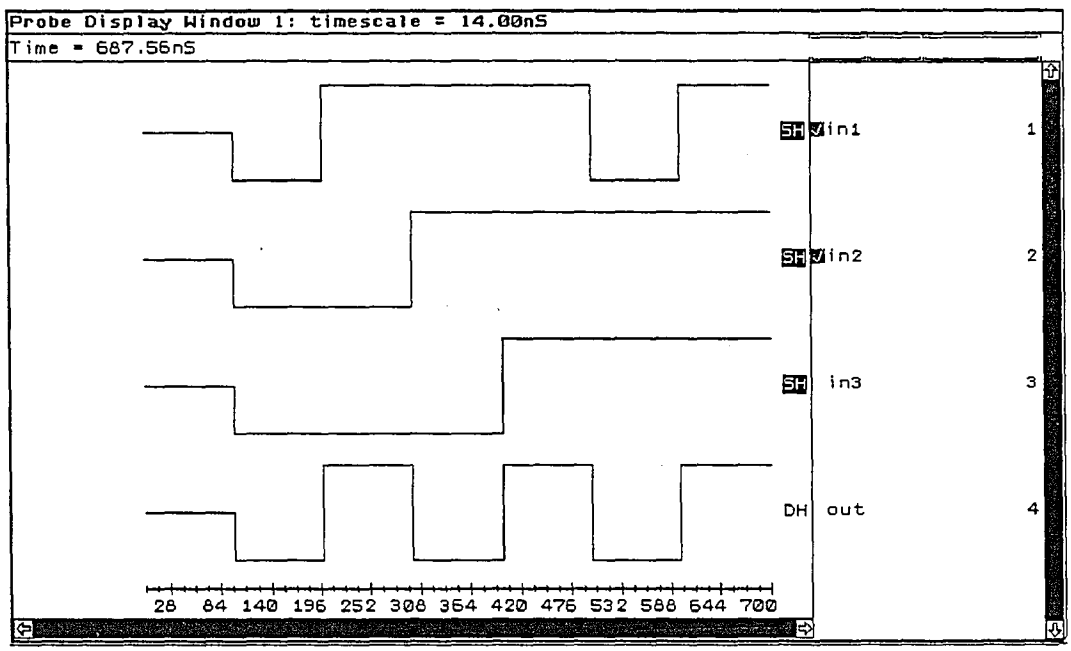


Fig 5.1 Simulation of a parity checker

Fig 5.1 shows that the model of the M language performs as parity checking. The above model is built directly from the operators. But for very large and complicated circuit, it is hard to write only by operators. There is another way to build up complicated model from several small cells by specifying the connections. The following example shows such kind a model of an adder which is built up by a parity checker and a majority voter:

```
MODULE fadder()
{
        IN LOGIC in1;
```

```
                IN LOGIC in2;
                IN LOGIC cin;;
               OUT LOGIC out;
    OUT LOGIC cout;
    BUILD {
    INSTANCE(parity_checking,parity_checking);
    INSTANCE(majority_voting,majority_voting);
/******          parity's outputs*******************/
    NET(parity_checking.in1,in1);
    NET(parity_checking.in2,in2);
    NET(parity_checking.in3,cin);
/******          majority_voting's outputs *******************/
    NET(majority_voting.in1,in1);
    NET(majority_voting.in2,in2);
    NET(majority_voting.in3,cin);
/******          final's outputs  *******************/
    NET(out,parity_checking.out);
    NET(cout,majority_voting.out);
    }
}
```

Fig 5.2 shows the simulation result of the adder. The statement "INSTANCE" in M indicates which subcell is used in this cell, and statement "NET" specifies the connection between the two items in it. With these two statements, we can build up a VLSI circuit.

3.Simulation of the design

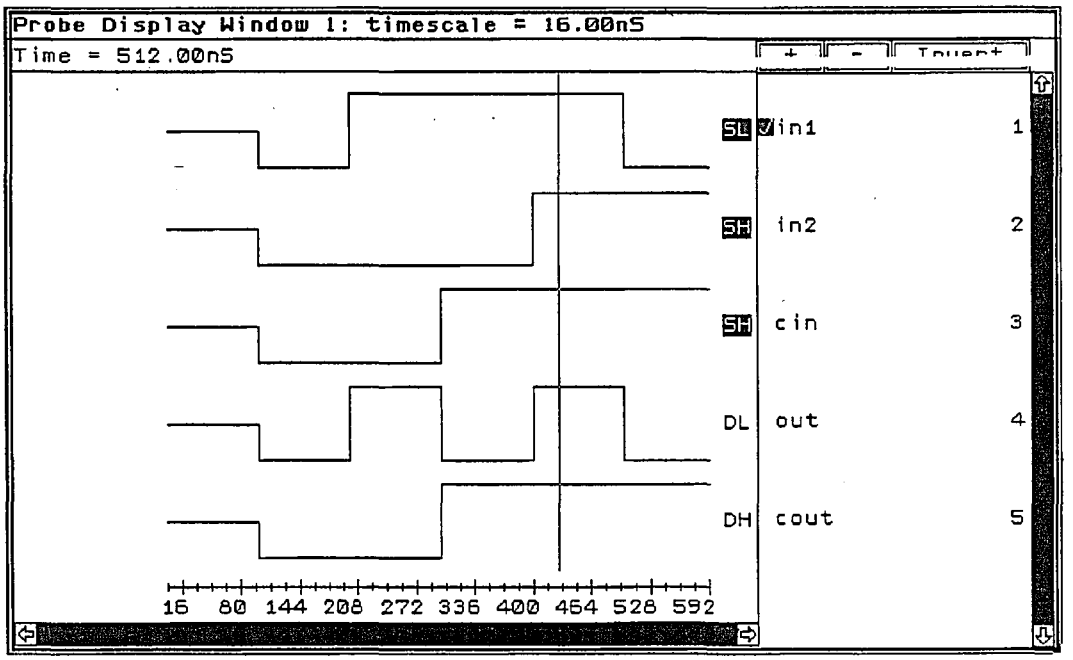


Fig 5.2 Simulation of an adder

Starting from very basic cells, the M model of this chip has about 80 different instances in M language finally. And the result shows that the design described in chapter 4 is correct. Among simulation, the design has been improved a lot, especially for rain flow part. And made us very clear about the connections of each small cells and timings of the circuit. Finally, the simulation of the design obtained the same result as the demonstration program did since we tested the simulation and demonstration program by same data files.

4. The hierarchy of the M instances

Fig 5.3 shows the hierarchy of the M instance. Besides those instances in the hierarchy, there are many general instances which can be called in any instance in the design. See appendix for more detail.

1. top.M : the top cell of the design
2. peak.M: instance to detect peak value
3. rain.M: instance to perform rain flow algorithm

4. count.M: instance to record and cumulate the histogram

5. clock1–4: generate the clock signals

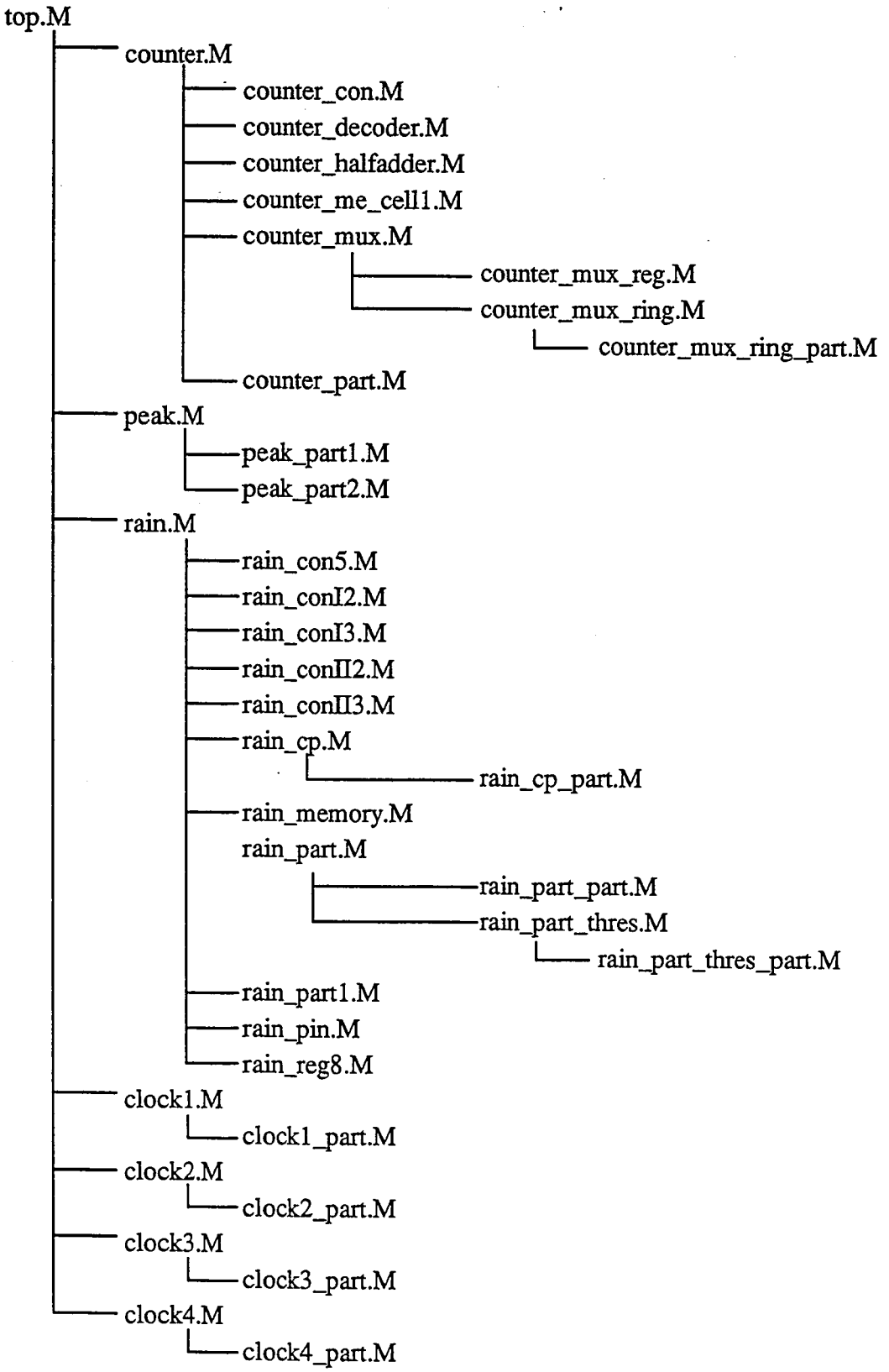


Fig 5.3 The hierarchy of M model

5.2. SIMULATION AT CIRCUIT LEVEL

For simulation at circuit level, we can substitute the small M instances by cells generated by Led(a GDT graphic tool) or directly written by a text editor. The following is a simplified netlist of an inverter generated by Led

```
N::
TECH scmos

CELL inverter
{
        VDD vdd0=1;
        GND gnd0=2;
        IN in0=3;
        OUT out0=4;
        TP tp0 w=400 l=300 g=3 s=4 d=1;
        TN tn0 w=400 l=300 g=3 s=2 d=4;
}
```

At this level as shown above, we can simulate the circuit at transistor level. In above netlist, 'w' is the the width of transistor, 'l' is the length of transistor. Other numbers indicate the node numbers which specify the connections among the p transistor , n transistor, input and output. Therefore we know the approximate size of every transistors in the circuit. We are going to use another CAD tool, magic, for layout design and another simulator Rsim to do the simulation of layout. Therefore the transistor size we can get from Lsim can not be consistent with Rsim. We have to determine physical size of each transistor in magic.

CHAPTER 6 SUMMARY AND FUTURE WORK

6.1. SUMMARY

At very beginning of this project, we spent a lot of time to understand the related civil engineering knowledge(such as the rainflow counting algorithm...)from professors and students of civil engineering department. Those discussions are very helpful to us. To verify that we have understood the algorithm correctly, we have developed and tested a software program using different data. We has used our own A/D conversion board to extract the actual data from the cassette tapes which recorded the real strain signal, and studied the signal, including frequency analysis and noise filtering. The final result shows that the software program can provide very close result to hand–counted result. Because the modify program will be used as part of the software to support the future system, it must have a convenient user interface for users. This was also a time–consuming job to write a interface of fully menu–hint and to support laser printer. Then we began to do the circuit design, wrote the circuit in M language and simulated it. While we discuss this project deeper with civil engineers, we have been changing a lot about the design, one big modification was to change the whole parallel design to serial design for high reliability and small size. This modification affected every aspects of the design. Another big change is to enlarge the memory size in order to support a long working period without refreshment. The current design can support continuous monitoring for at least one year under normal traffic if the chip is applied to a bridge.

6.2. FUTURE WORK

The ideal future system as shown in Fig 3.1 requires a lots of work.

1. Chip testing

Chip testing requires development of software program to communicate with the chip, to read data in and write command out . Future system will also be supported by this function of the program. And testing result should be consistent with the expected result, otherwise any fault must be checked out in chip design.

2. Sub system

A sub–system including strain gauge, signal conditioning circuits, A/D convertor, digital filter and the fatigue data processing chip will be built up and tested in the DSP(digital signal processing) laboratory. A sample strain signal must be constructed and be used to test the sub–system.

3. Test the sub–system in the ATLSS Laboratory

The sub–system will be tested using real strain signals from the tested structures in the ATLSS lab. We will modify the design based on the feedback from the laboratory testing.

4. Technology transfer

We will consider many real problems, like the estimation of power consumptions, electric shielding, and fabricate the whole circuit to a small module.

5. Test the sub–system in the field

The sub–system will be tested on bridges , highways , railways ... and reports problem to modify the sub–system.

6. Incorporate communication hardware with the sub–system

Though there are many other ways to carry out the communications between remote modules on tested structure and central control, the ideal one is by radio signal. The communication can be achieved by a radio signal modem chip. Its function is the same as normal modems but not connected by telephone lines. Our research group has done some

works on it. If we can use the radio modem chip, then user will be able to remotely control the sub–system on structure.

REFERENCES

1. Ben T.Yen, et al, *"Manual For Inspecting Bridges For Fatigue Damage Conditions"*, Department of Civil Engineering Report No. 511–1, Lehigh University, 1990.

2. Hugh M. Woodwar, John W.Fisher, *"Predictions of Fatigue Failure In Steel Bridges"*, Department of Civil Engineering Report No. 386–12, Lehigh University, 1980

3. S. D. Downing and D. F. Socie, *" Sample Rainflow Counting Algorithms"*, INT.J.FATIGUE January, 1982

4. Alan V.Oppenheim , Ronald W. Schafter, *"Discrete–time Signal Precessing"*, Prentice–Hall, Inc, 1989

5. Neil Weste, Kanran Eshraghian, *"Principle of CMOS VLSI Design"*, AT&T Bell lab, 1985

6. Leland B. Jackson, *"Signal ,System and Transform"*, Adderson Wesley Publishing Company, Inc., 1989

7. *GDT manuals*, Silicon compiler Systems Corporation 1989

APPENDIX

A list of instances used in M model

INSTANCES	DESCRIPTION
Fcomparator.M	A 12 bit serial input comparator
Fcomparator_reg2_phi1.M	A sub instance for register clock control in Fcomparator
In2reg.M	A 2 parallel input register
PtoS1.M	A parallel to serial converter
PtoS1_switch.M	A switch for output of PtoS1.M
PtoS_part.M	A sub instance for register clock control in PtoS1.M
Qreg.M	A qualified parallel register
Qsubtractor.M	A qualified sequential subtracter
Qsubtractor_part.M	A sub instance for register clock control in Qsubtractor
clock1.M	A clock generating Phi1ss and Phi2ss
clock1_part.M	A sub instance of numbers of and–gates
clock2.M	A clock generating Phi1s and Phi2s
clock2_part.M	A sub instance of numbers of and–gates
clock3.M	A clock generating Phi1b and Phi2b
clock3_part.M	A sub instance of numbers of and–gates
clock4.M	A clock generating Phi1 and Phi2
clock4_part.M	A sub instance of numbers of and–gates
counter.M	the counting circuit
counter_con.M	control circuit
counter_decoder.M	A 5 bit input decoder
counter_halfadder.M	A half adder
counter_me_cell.M	memory cell in couting circuit
counter_mux.M	mutipliexer in counting circuit
counter_mux_reg.M	A special 64 input register
counter_mux_ring.M	An array of 32 2 input registers in serial
counter_mux_ring_part.M	A sub instance for register clock control
counter_part.M	A sub instance for register clock control
fadder.M	A 1 bit full adder

gnd.M	ground
inverter.M	A inverter
or.M	A multi–input or–gate
or2.M	A 2 input or–gate
peak.M	Peak circuit
peak_part1.M	A sub instance for register clock control
peak_part2.M	A sub instance for register clock control
rain.M	Rain flow circuit
rain_con5.M	A sub instance for register clock control for register 5
rain_conI2.M	A sub instance for register clock control for register 2
rain_conI3.M	A sub instance for register clock control for register 3
rain_conII2.M	A sub instance for register clock control for register 2
rain_conII3.M	A sub instance for register clock control for register 3
rain_cp.M	CP circuit
rain_cp_part.M	A sub instance for register clock control for register
rain_memory.M	Memory circuit
rain_part.M	Decision circuit
rain_part1.M	A sub instance for register clock control for register
rain_part_part.M	'ctrl' circuit in decision circuit
rain_part_thres.M	'threshold' circuit in decision circuit
rain_part_thres_part.M	A sub instance for register clock control for register
rain_pin.M	'pin' circuit
rain_reg8.M	register 4
reg.M	A multi–input register
reg1.M	A 1 input register
reg_2_input.M	A 2 input register
reg_3_input.M	A 3 input register
reg_4_input.M	A 4 input register
regm_12.M	An array of 12 1 input registers in serial
top.M	top instance
top.i	command file for simulation
vdd.M	Vdd

VITA

The author was born in China on Feb 9, 1966. He received his Bachelor Degree of Science in Applied Physics from Tsinghua University, Beijing, in 1989. Since fall of 1990 he enrolled in the Department of EECS, Lehigh University and studied in VLSI signal processing.

END

OF

TITLE